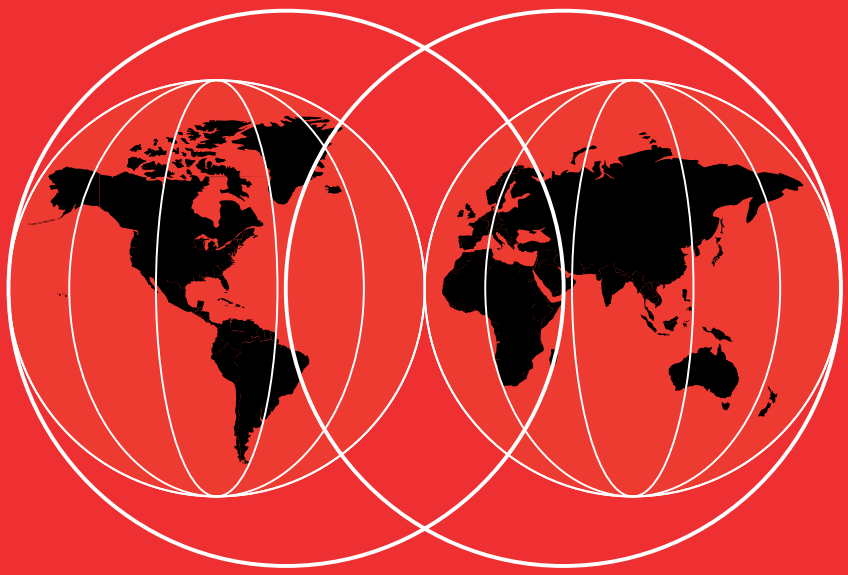
Lotus

IBM

Guide to Deploying Domino Go Webserver

Maurizio Barazzuol, Fiona Collins, Rudolf Jetzelsperger, Michael McGuire



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-2002-00

International Technical Support Organization

Guide to Deploying Domino Go Webserver

March 1998

Take Note!
Before using this information and the product it supports, be sure to read the general information in the Special Notices section at the back of this book.

First Edition (February 1998)

This edition applies to Release 4.6 of Domino Go Webserver.

Comments may be addressed to: IBM Corporation
International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Preface

This redbook describes Domino Go Webserver and provides information on how to deploy the product in a number of different environments.

The earlier chapters of the book introduce Domino Go Webserver architecture and describe some of the planning and installation considerations relating to the product. Later chapters discuss the security, administration and performance features of Domino Go Webserver. The performance chapter describes some common tuning tips to help improve server performance.

Some of the application programming services of Domino Go Webserver are documented, along with a summary of those services provided by a number of related products.

In the final chapter of the book, a method for determining the appropriate Lotus Domino server product to host the different types of Web application is described, along with a discussion of the considerations for moving content between the Domino Go Webserver and Domino.

The redbook was written for technical specialists and programmers in customers, IBM business partners, and the IBM and Lotus community, who need a good technical understanding of how to deploy Domino Go Webserver.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Cambridge Center.

Fiona Collins is a Lotus Notes / Domino Specialist at IBM's ITSO Center at Lotus Development, Cambridge, Massachusetts. She manages projects whose objective it is to produce redbooks on all areas of Domino. Before joining the ITSO in 1996, she provided technical support for Lotus Notes and the AS / 400 for Lotus and IBM in the UK.

Maurizio Barazzuol is an independent consultant who works with Dator s.r.l., an IBM and Lotus Business Partner in Bolzano, Northern Italy, and IBM Italy Education & Training. He graduated in Physics at Padua University, Italy. He provides technical support for systems integration projects in the mid-range systems environment, develops CBT courses, and gives classes and lectures.

Rudolf Jetzelsperger is a Senior Consultant with IBM Global Services in Boulder, Colorado, USA. As an architect with the Object Technology Practice, his main role is the development of strategic e-Business solutions for IBM customers. Rudolf has over 11 years experience in consulting with client organizations in the insurance, distribution, defense, travel, technology, telecommunications, process, and manufacturing industries.

Michael McGuire is a Senior I/T Specialist at IBM Global Services in Chicago, Illinois, USA. As a systems specialist with Networking Services, his primary responsibility is Domino network design and performance analysis. Michael has over 12 years experience with various types of networking technologies.

A number of people have provided support and guidance throughout the production of this book and we would like to that the following people in particular (in alphabetical order):

- Charlie Bradley, IBM Raleigh HTTP Performance Team
- Chris Conway, ITSO Poughkeepsie
- Don Harbison, Notes/Domino Product Manager for Application Development, Lotus Cambridge
- Michele Pennell, Internet Marketing Manager, Lotus Cambridge
- Jon Rush, IBM Rochester
- Carla Sadtler, ITSO Raleigh
- Graphic Services, Lotus North Reading

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us your comments about this, or other redbooks in one of the following ways:

- Fax the evaluation form found at the back of this book to the fax number shown on the form.
- Use the electronic evaluation form found on the redbooks Web sites:

  For Internet users `http://www.redbook.ibm.com`

  For IBM intranet users `http://w3.itso.ibm.com`
- Send us a note at the following address:

  redbook@vnet.ibm.com.

Chapter 1
Architecture

Domino Go Webserver and the Network Computing Framework (NCF)

Domino Go Webserver

Lotus Domino Go Webserver is a scaleable, high-performance Web server that runs on a broad range of platforms that has evolved from the IBM Internet Connection Secure Server (ICSS), version 4.2. Domino Go Webserver provides the basic services to deploy a Web site and a network computing application environment and includes enhancements that are not available under ICSS:

- The Java Virtual Machine and other services required to run Java servlets.
- The runtime environment for the NCF programming.
- The infrastructure that uses standard LDAP directory and SSL security protocols.
- An ORB to provide IIOP access to CORBA objects, for example objects activated in a Component Broker Connector environment.

Lotus Domino Go Webserver is a complete Web server implementation; however, it supports the NCF programming model runtime environment other vendor's Web server implementations. It provides NetQuestion search capabilities to help users find information quickly. It also offers Web Usage Mining to track the frequency and types of hits to sites. Domino Go Webserver is a foundation of the IBM / Lotus NCF.

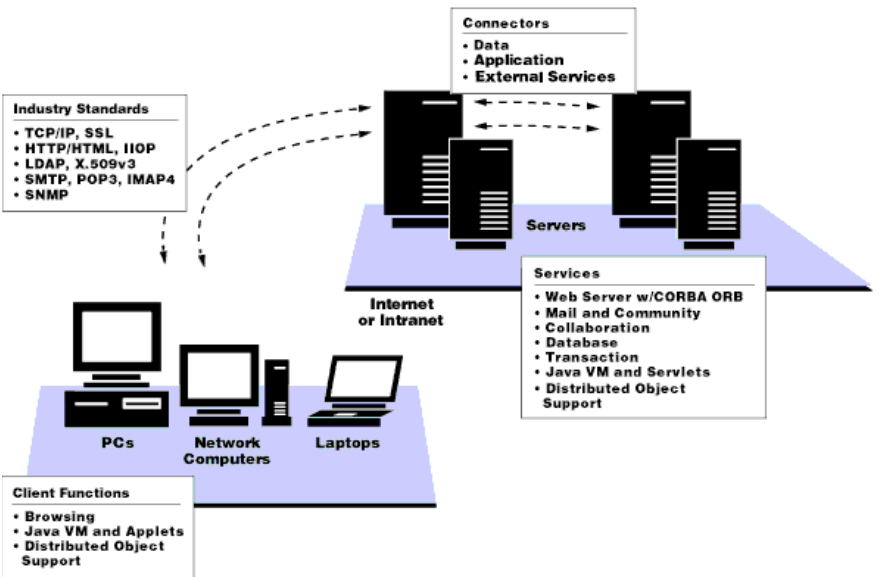
IBM/Lotus/Tivoli Network Computing Framework (NCF)

The Network Computing Framework is an architecture created to help customer and industry software provider development teams design, develop, deploy and manage e-Business solutions across the enterprise. The NCF contains six key elements:

1. An infrastructure and a set of services whose capabilities can be accessed via open, standard protocols and a standard component interface, JavaBeans.

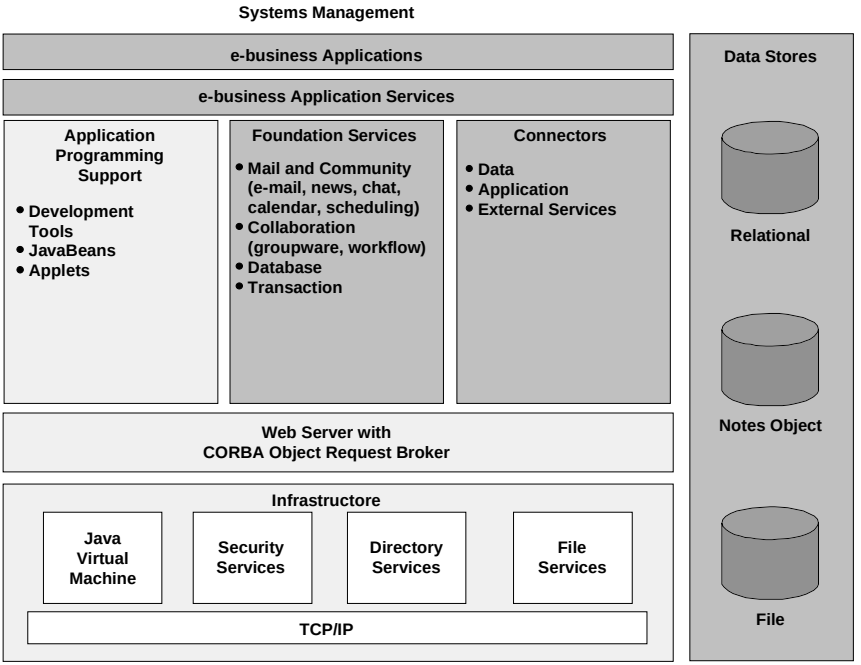
2. Clients based on a Web browser Java applet model that support universal access, a thin client paradigm, and exploitation of "just in time" delivery of components to provide a rich user interaction with the server.
3. A programming model and tools that create and exploit JavaBean components to build robust e-Business applications. As a result, any tool can produce a component to access any service.
4. Internet-ready protocol support, such as HTTP and IIOP, that links JavaBean components.
5. A set of "connector" services that provide access to existing data, applications, and external services.
6. A set of built-in collaboration, commerce, and content services that provide a foundation for an industry of partner-built solutions and customizable applications for e-Business.

The figure below illustrates the NCF model of a prototypical network computing solution. It is a logical three-tier model designed to support thin clients with high-performance Web and enterprise servers.



Domino Go Webserver Logical Subsystems

The following diagram describes the NCF architecture. The highlighted areas are the subsystems that are provided by Domino Go Webserver. The subsystem services are described in more detail in the following sections:



Web Server with Object Request Broker

As the foundation for the NCF, the Web server provides the following services:

1. It is the entry point to server functionality supporting HTTP and IIOP requests from NCF clients.
2. It supports the mechanisms for locating and invoking business logic on the logical mid-tier server.
3. It provides the NCF programming environment for writing robust, transactional business logic applications.
4. It provides the linkage between the Web and existing enterprise applications, data, and systems.
5. It coordinates, collects, and assembles Web pages composed from static and dynamic content and delivers them to NCF clients.
6. It operates based on an understanding of user preferences.

7. NCF's standard interfaces and protocols allow substitution of any vendor's Web server implementation for example, Web servers produced by IBM, Microsoft or Netscape.
8. The Object Request Broker incorporated in the Web server supports IIOP to allow JavaBean components to communicate client to server, and server to server to satisfy a user's request. This enriches the environment provided by a simple Web server. For example, it allows multiple to work simultaneously and asynchronously on a single request. One may check inventory, another check credit, and yet another provide news or promotional material on related areas.

Infrastructure

The NCF infrastructure provides the underlying support for the Web Server and Object Request Broker subsystem. It includes the security, Java Virtual Machine, file, directory, and TCP/IP services, all based on open standards.

Security Services

Security services are functions performed or provided for, within any given system environment, that are expected to support and enforce the security policies defined as part of an organization. The following are the security services and the mechanisms that support them in Domino Go Webserver.

Identification and Authentication

These services allow the verification of the identity of the individuals and applications using your systems and networks. Three types of entity authentication protocols are supported.

1. "One party" authentication is commonly used when users log on to a system, network or application. Secret information such as a password, is known to the user and the resource being accessed.
2. "Two party" authentication is generally used when a distributed application communicates with its other parts, as when systems join in a communications network. Secret information, such as a cryptographic key, is shared among all parts of the distributed application.
3. "Three party" authentication is typically used by 2 different applications which prefer to use a trusted third party (often called a certification authority) rather than share secret information as required in two-party authentication.

Domino Go Webserver uses SSL (version 3) to perform two- and three-party authentication.

Access Control

These services let you provide the desired level of protection to your information resources in all forms, wherever they reside. There are 2 mechanisms commonly used to provide access control: access control lists

and security labels. Access control lists contain information about which users or programs can access the resource associated with the list. They also describe the type of access permitted for the user or program (for example, read, write, and execute). Security labels are used to classify resources according to your defined security policy. To gain access to the resource, a user's or program's authority needs to match the security label content of the particular resource.

Confidentiality

These services enable you to protect valuable information from unauthorized disclosure. These services are generally provided by encipherment / decipherment facilities like the Data Encryption Standard (DES). Domino Go Webserver uses SSL (version 3) to provide the confidentiality services.

Data Integrity

These services allow you to safeguard critical data or programs from unauthorized modification. These services are generally provided by modification detection and message authentication mechanisms. Domino Go Webserver uses SSL (version 3) to provide the data integrity services.

Non-repudiation

These services let you prove the origin or delivery of a message or data. Digital signature facilities are used to provide this service. These mechanisms use the Rivest, Shamir and Adleman cipher (RSA) public key algorithm to uniquely tag the selected information. Once tagged, the origin of the information can be repeatedly verified. Similarly the information that was sent can be verified to arrive at its destination without having been changed.

Java Virtual Machine Services

The Java Virtual Machine in both clients and servers provides the base support for programming in the NCF and provides platform independence for e-Business solutions.

File and Directory Services

At the highest level, information stored in the distributed directory is used by a resource manager to locate the target server (for the requested resource), select the appropriate protocol for communication with the target, and bind the appropriate method or driver code to the logical connection. The process of resolving the federated name of the resource returns the network location (network name) and communication / transport protocol (binding information) necessary for communication with the target server. This information (such as LU name, mode name, and TP name) is then used as input to Network Services to establish the appropriate transport connection and binding and to complete the binding of the required communication method or driver code for the logical connection.

Directory services that locate users, services, and resources in the network are accessed using industry-standard LDAP protocols and interfaces. Java developers can access these services through the JNDI programming interfaces.

File services extend the Web server to support scaling to large environments by providing transparent access to geographically dispersed files. Files services provide a uniform hierarchical name space, high performance, and secure access to enterprise files.

TCP/IP Services

Requests are routed among clients and servers using TCP/IP. Secure Sockets Layer (SSL) is supported for application security and IPSec is supported for network security to provide data encryption, client/server authentication, and message integrity. TCP/IP is extended to allow for the creation of virtual private networks that provide secure communications across the Internet and to provide connectivity across the leading wireless packet data and cellular networks.

Routing mechanisms allow a single Web site to be supported by multiple computers that share the workload. These mechanisms allow Web sites to scale to handle a high volume of interactive requests from a large number of clients.

Application Programming Support Services

This subsystem describes the programming support services provided by Domino Go Webserver. These services are categorized as follows: development environments, programming interfaces, and analysis. The Domino Go Webserver tools to support these services are described in more detail in the Performance and Development chapters.

Development Environments

Development environments enable a Web site developer to create and maintain application logic. Enhanced tools speed the development and quality of the application that are developed. Domino Go Webserver includes the Java Development Kit (JDK) version 1.1 and an extended CGI development environment. Domino Go Webserver Pro augments the base product's development tools with NetObjects Fusion for building and managing a Web site, and Lotus Bean Machine for creating Java applets/servlets using JavaBean technology.

Programming Interface

Programming interfaces provide a mechanism for application developers to enhance the standard Web features by providing direct interfaces into the Web server. Domino Go Webserver provides a mechanism for accelerating performance of a Web site through the use of Go Webserver Application Programming Interface (GWAPI).

Analysis

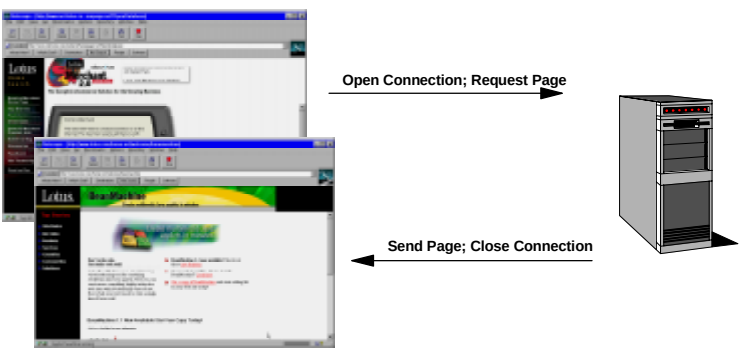
Analysis services provide mechanisms for analyzing the characteristics of the Web site. Characteristics include development, performance and usage of the Web site. Domino Go Webserver provides tools for analyzing the site content (using the Platform for Internet Content Selection (PICS) standard), server activity, and the site usage.

Web Site Models

In this section we will define some prototypical Web site models to be used throughout the book that will show how Domino Go Webserver options relate to different types of Web sites. These models are not meant to represent all possible Web sites but will be representative of the major Web site alternatives. We use two dimensions to define a Web site models. One dimension is how the user of the Web site interacts with the Web site. The other is the type of network that the Web site supports. These are defined as user interaction models and network models. A Web site model is a combination of an interaction model and a network model. The implications of these Web site models will be discussed throughout the book

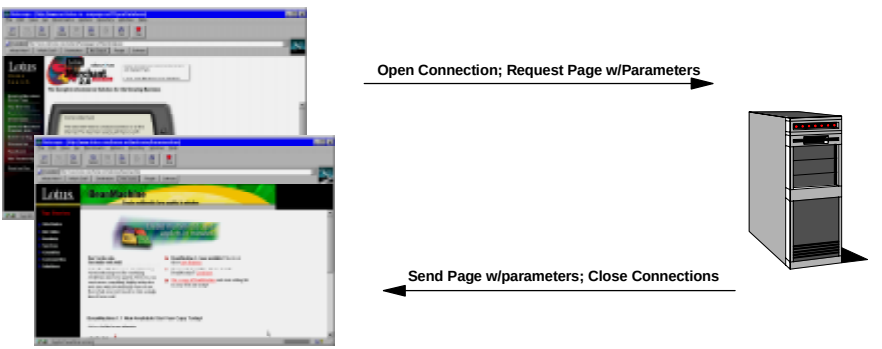
Simple User Interaction Model

The simple Web model consists of publishing static Web pages using the Hypertext Markup Language (HTML). The Web browser opens a connection to the Web server and the server returns a page and closes the connection. This model is suitable for providing access to business information that is relatively stable. It originally provided the foundation for the Web. Although later enriched by extensions to HTML (such as tables and frames), the simple Web model remains a static model. The user's only interactive options are limited to clicking on a link to visit another page or requesting a page reload.



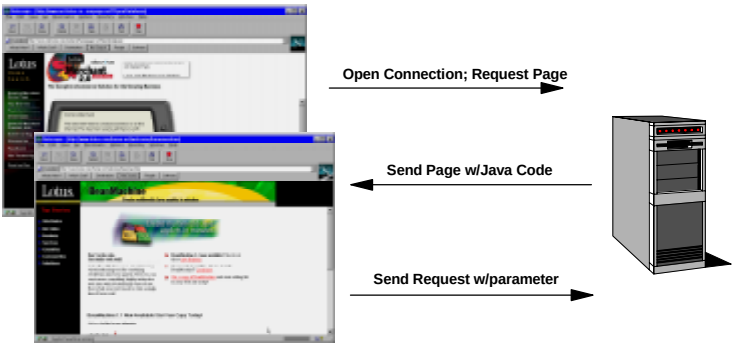
Interactive User Interaction Model

In the interactive Web model, pages can contain forms, fields, and buttons that allow users to enter data and choices. When users complete the forms and make their choices, the browser connects to the server to allow the data and choices to be transmitted. The Web server passes the information to a custom server program or script that makes an inquiry or does calculations and passes back a new page for the browser to display. The connection is then closed. The interactive Web model supports a simple form of client/server computing using Hypertext Transfer Protocol (HTTP) as the middleware and the Web server's Common Gateway Interface (CGI) that calls a custom server program or script. This model works well for simple interactions that do not require high inter-activity between the client and server. However, it is quite expensive in terms of server resources and time when establishing a connection between the browser and the server for each interaction.



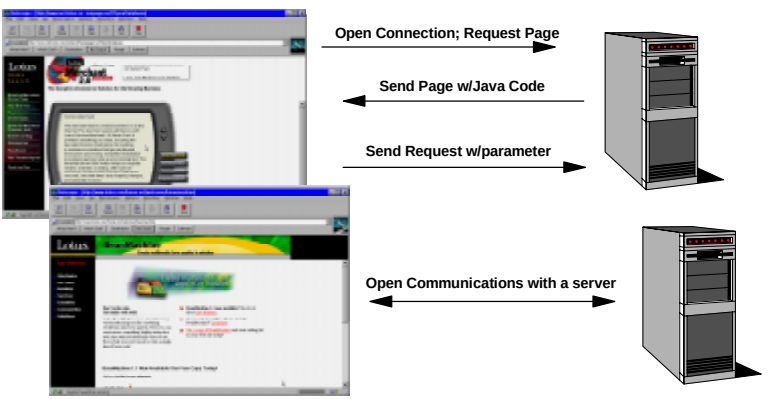
Distributed User Interaction Model

The distributed Web, also known as Internet PC, introduces Java. With Java, a small application called an applet can be transmitted to the browser along with the Web page. The Java applet runs within the browser as users view the page. It can provide animation or sound as well as rich graphical user interface components such as the ones usually found on windowed PC applications. The applet can display a window or be displayed in a Web browser; it can also process user input. If server interaction is needed, the applet can open a connection to the Web server to access a server or script through the Web server's CGI. The distributed Web model enriches the user interface and off-loads tasks from the Web server (such as parameter checking in the browser), but it is still tied to HTTP for communications. The next section describes another model that supports additional communication protocols.



Enterprise Distributed User Interaction Model

An additional application model that IBM supports is called the Enterprise Distributed Web model. Enterprise Distributed Web applications are true open client/server applications. The client is coded in downloadable Java, which runs in any Java-capable browser. A page containing the Java applet is downloaded by a Web server to a browser. While the applet runs in the browser, it opens its own communications session to the server. The applet also communicates with a server to provide access to the database, C++, or Java application servers, in addition to supporting a variety of middleware, include HTTP, TCP/IP, Secure Sockets Layer (SSL), DSOM, and MQ.



Intranet Network Model

An intranet is a network that generally resides within an organization. One of the major differences between running applications on the Internet versus the intranet and the extranet is security (or the lack of it) and what kind of information is presented to the user. While using the intranet, all information

is available to the users with the right authority (for example; a sales person, to all sales and customer related data; a bookkeeping person, to all personnel and human resource data, and so on).

Extranet Network Model

An extranet is a network that spans multiple organizations but is not open to the general public. The extranet can be viewed as a private net within the Internet or a private network using either dial-up or leased lines with its own security and encryption algorithms. So data exchanged between organizations is relatively safe. However, here it is important to have a mechanism in place to make sure that when you send something, the addressee knows exactly who sent it and also keep a receipt for delivery to have proof of sending it.

Internet Network Model

The Internet is the public network that is accessible to the general public. On the Internet everything should be protected except information to be used by the public. Under more restricted access, users are allowed to see and modify specific data, but always with the exposure of accidentally disclosing this information also to the public (hackers).

Web Site Model Matrix

The following matrix describes the various Web site models that will be referenced in this book. The columns represent the network model and the rows represent the user interaction model.

Model	*Intranet*	*Extranet*	*Internet*
Simple	Simple Internet Web Site	Simple Extranet Web Site	Simple Internet Web Site
Interactive	Interactive Internet Web Site	Interactive Extranet Web Site	Interactive Internet Web Site
Distributed	Distributed Internet Web Site	Distributed Extranet Web Site	Distributed Internet Web Site
Enterprise Distributed	Enterprise Distributed Internet Web Site	Enterprise Distributed Extranet Web Site	Enterprise Distributed Internet Web Site

Summary

This chapter:

- Described the logical subsystems and functions that make up the Domino Go Webserver product
- Positioned Domino Go Webserver against the background of the Network Computing Framework Architecture
- Outlined our Web site matrix as a model for classifying Web site applications and installation requirements.

The following chapters in the book will expand on the functions and capabilities of Domino Go Webserver and on possible product evolution strategies for Domino Go Webserver installations.

Chapter 2
Installation Considerations

Rather than cover the installation process for Domino Go Webserver, step by step, this chapter will describe some of the installation planning activities for deploying the product.

Domino Go Webserver is available on many software platforms and every environment has its own hardware and software requirements and installation procedures. If you need technical information about a specific installation process or the system requirements, please refer to the Domino Go Webserver Quick Beginnings manual for the relevant platform.

Installation Decision Points

In this section we will document some issues to be considered as part of the installation planning process. Whatever the size of your installation, certain basic planning activities need to be undertaken.

The development of new Internet standards and the emergence of new applications make every detailed list incomplete. This book is not a planning theory manual; our goal is only to highlight the importance of a pre-installation phase when planning a new Web site and to point out some of the issues involved.

Application Dependent Remarks

As in all other processes of enterprise information management, look first at your applications. In order to develop effective Web applications you must do more than the traditional application analysis. In the previous chapter we suggested a matrix to classify Web site models; you could use that matrix to classify your applications in the World Wide Web environment.

As a simple exercise, take that 4 x 3 matrix and mark the square where you think your applications best fit. Perform this action twice:

1. For the applications you are currently planning to develop or migrate to the Web, and
2. For the applications you plan to develop or migrate to the Web in the next year. Be imaginative: last year you were probably not thinking of a Web server installation like the one you are planning now.

The two matrices will probably be very different. Use them, or any other model you prefer, to understand which kind of Web site you are going to build and your future needs.

The Development chapter, later in this book, can help you understand some of the Web-dependent technical aspects of the type of application that you are planning, and to evaluate carefully all software requirements.

Which Platform to Choose?

Many factors will affect the choice of platform that Domino Go Webserver will be installed on. These factors will work together to define the constraints that Domino Go Webserver will operate under and may well affect the functionality of your application to some extent.

The first, and often overriding, factor is the budget.

Within this, even in the simplest Web site models (static Web) the main element you need to consider is performance: you will need to have machines and software that support the expected number of active users accessing your application.

In extranet and Internet solutions, Web availability is also an important factor in that your users may live in different time zones, have different holidays and so on. Also, your users may not all work during the same hours; some may work during the night, or during weekends. The degree accessibility and flexibility is a great strength of a Web application, but it does mean further planning is necessary for the applications developer and IT department.

In an intranet solution, availability is also an important constraint but perhaps not so critical as in the other two environments.

Moving towards the more complex Web models, software and data availability requirements become gradually more crucial and more complex.

Do you have an existing application to share with Web users?

Do you have a current production database and a new Web application which must update its data?

In Chapter 4, Development, we describe some of the add-in products available that can help to answer these kind of questions, but not every solution is available on every platform and you will need to check all software requirements carefully.

Client Requirements

An important step in application development planning is defining the client requirements.

For example, in a client/server environment you may determine that you need a Windows 95 client with X Mb of main memory, or a Windows NT Workstation Client with Y Mb of main memory, and for both to have Z Mb of free hard disk space available. The need to define these client resources does not apply for extranet or Internet solutions since the client is a browser. However, you will still need to decide which browser to use and to define the workstation to be used (even though the resources required will be less critical).

You may also want to consider whether users accessing the application will be using different browsers, with differing functionality and the effect that this may have on the performance of your application. Domino Go Webserver supports browser detection, which allows you to customize your server to provide content tailored for the type and version of the client browser being used. Although the most popular browsers support modern Web features, you cannot be sure that all your clients are using up-to-date versions of the browser. Providing content tailored to the most popular browsers used to access your Web server will increase the attractiveness and effectiveness of your site.

Also, consider whether you will need to support multiple languages and the potential code page issues that may result.

Networking Considerations

Whatever Web model your application fits, you will need to provide Internet connectivity and therefore the speed of this connection is critical. Your connection should not be any slower than 56Kbps (64Kbps in Europe). Remember that it is not only the speed of the connection from your server to the ISP that is important, but also the speed of the connection from the ISP to the next node in the Internet. Many ISPs also offer other services besides a simple connection. You will probably want to select an ISP based not just on the speed of the leased lines that they offer.

You must define a domain name for the Web, one that can be easily remembered by extranet and Internet users and that reflects your company name and the IP addresses you need for the connections.

For performance and security reasons, it is very important that you understand your internal network topology. You will need to have complete control over the network routing path in order to maintain a high level of security and to control any performance bottlenecks.

Also, for security reasons all external non-Internet connections should be carefully controlled.

Enterprise Security

Much, perhaps too much, has been written about security in the Internet environment. It is right to worry about security, but nowadays all the tools we need are available to build a secure Web site. Internet security is not the problem that it once was. Chapter 3 of this redbook explains some of the options available with the Domino Go Webserver product to provide a secure Web site.

In the planning phase, bear in mind when evaluating the security policy for your Web application that a comprehensive enterprise security policy may consist of several different security levels: at the server, network, application and user level.

Multiple Web Servers

It is possible to run multiple instances of Domino Go Webserver on one system. You can also have multiple virtual servers hosted on a single Webserver or a combination of both.

Note Depending on the platform that you are running, there may be some limitations to the number and functions that can be achieved.

You may want to run multiple copies in order to obtain the following benefits:

- To limit server administration requirements.
- To control or limit access to certain data or applications.
- To control the system resources allocated to each server.

As examples, you could assign different server environments:

- For different areas of the business. For example, you could have a static Web server for a public marketing application, but a more secure server for an interactive commerce application.
- For different applications. Install intranet applications on one server and extranet applications on another server.
- To separate production and test servers, so that updates are not made to the production server until they are fully tested.

Chapter 6, Administration, describes some examples of multiple server configuration.

Web Users and System Users

Web users are those users that are defined in the Web server. You would normally have:

- At least one explicit user defined in your Web server: the Webmaster
- Another implicit user defined, for example a Guest or Public user.

System Users are those users defined to the Operating System.

The distinction between these two types of users is important when it comes to the security of your Web server. When Domino Go Webserver runs, it uses a specific System User authority that allows it access to all the system objects allowed for that authority. This means that a generic, unrestricted, Web user can potentially access all the objects that the Web server has the authority to.

Web server configuration directives can grant specific authority levels to a single Web user or to a group of users. This type of security access can be new to application developers not used to developing applications for the Web environment and will need some planning to make sure that it is effective.

Web Directory Structure

The definition of the directory structure for a traditional application (that is, a non-Web enabled application) is usually undertaken by an application analyst or a senior programmer. When working with a Web server application the directory structure is much more important because it also includes URL redirections. URL redirections enable a browser request or a CGI program call to be transferred to another Web server. This is a common function in Web applications.

The Web directory structure may also be important for the following reasons:

- Security planning
- Server management and administration
- A good Web directory structure can make an application programmer's job much easier and less complex.

The design of the Web directory structure needs to be performed at the start of the development process and ideally, the person responsible for its definition is knowledgeable in resource mapping concepts and commands.

Planning for the Installation: A "Simple Environment" Example

We will use this simple example to illustrate some of the issues that we have highlighted and point out some of the decisions that need to be made. Imagine a very small business enterprise with a Windows NT server installed. They have a working knowledge of the operating system but are not experienced in Web server application development. What questions should they be asking?

Which NT? On Which machine?

You may install a Domino Go Webserver on either a Windows NT server or a Windows NT workstation. A Windows NT server may have the advantage in terms of availability and security capabilities, but a Web server exposes the machine and the application to external users, and so there are additional security considerations.

If external users do not need to access your server or applications, installing a Domino Go Webserver on a Windows NT workstation may be the cheapest and safest solution.

The costs and benefits of the following should also be added to the equation:

- The advantages and disadvantages of a single machine environment vs. a multiple server environment for both hardware and software (remember that you may require specific additional software to connect different servers in the best way!)
- The levels of security, availability and performance that you need in the different environments, including the Web server and other server functions that you will need: file serving, printer serving, database serving, etc.

Which File System?

Windows NT, like other operating systems, supports different file systems, each with distinctive security and performance features, file types supported and so on. You can install the Domino Go Webserver on a HPFS or NTFS formatted drive.

Note We advise you to use an NTFS formatted drive, because it will allow you to use the Windows NT security features.

Normal Application or Windows NT Service?

This can be an important question as you can run Domino Go Webserver as a normal application or as a Windows NT service.

To start the Domino Go Webserver as a normal application, you have to log on as a user, start the server and remain logged on as long as the server is to be up. As a consequence:

- The server is running with the authority of the user that started it.
- No other user may log on to the machine while the server is running.

You may choose to start the Domino Go Webserver as a Windows NT service through the Services icon in the Control Panel. As with all Windows NT services, you can then launch the server when the system is started up, automatically. The Domino Go Webserver will run in the background with its own authority without affecting any other activity on the server.

Note Only one instance of Domino Go Webserver may be run as a Windows NT service. Other instances have to run as normal applications.

Installation Tips

Installing Domino Go Webserver on a Machine That Has Previously Had ICSS Installed

If you are installing Domino Go Webserver on a machine that has previously had ICSS installed on it, you must de-install ICSS via the ICSS install tool before installing Domino Go Webserver. If you don't, your system will contain pieces from both ICSS and Domino Go Webserver. The Go Webserver install tool does not sense the presence of a prior release of ICSS so you can not use that to de-install.

Using Domino Go Webserver 4.6.1 as a Proxy Server

If you are using Domino Go Webserver 4.6.1 as a proxy server you must manually change the httpd.conf file to select the protocols for which the server will act as a proxy. Currently the Configuration and Administration forms page under Proxy Server Settings allows you to choose the protocols: HTTP, Gopher, FTP and SSL (https). Even if you select these and issue Apply, the configuration file will not be updated correctly.

The unedited httpd.conf file will be:

```
Pass http:*

Pass gopher:*

Pass ftp:*

Pass https:*
```

These need to be changed to:

```
Proxy http:*

Proxy gopher:*

Proxy ftp:*

Proxy https:*
```

In other words the Pass directive needs to be changed to Proxy in order to select the protocols for which the server will act as a proxy.

This manual change is only required for Domino Go Webserver 4.6.1 as Domino Go Webserver 4.6 used the Pass directives so it was not affected, and future releases will have this change already coded in them.

Installation Models Matrix

In this chapter we have touched on some of the issues involved with planning and installing Domino Go Webserver. Finally, the following table uses the same criteria as we have used in several other chapters, in terms of type (complexity) of Web application and the environment/audience for that application, as we summarize the main issues and planning decisions that you will need to focus on in each environment.

The + or - sign indicates the relative importance of the item in any particular environment. The planning topics and issues are:

- Applications Analysis
- Availability
- Client requirements
- Multiple hosting and multiple servers
- Network topology
- Security
- Software requirements
- Web directories

Models	*Intranet*	*Extranet*	*Internet*
Simple	• Security (-) • Web directories(-) • Network topology(-) • Availability(-)	• Security (+) • Client Requirements • Multiple hosting • Web directories • Network topology(-) • Availability	• Security (+) • Client Requirements • Multiple hosting • Web directories • Network topology • Availability
Interactive	• Security (-) • Application analysis • Web directories • Network topology(-) • Software requirements • Availability(-)	• Security (+) • Client Requirements • Multiple hosting • Application Analysis • Web directories • Network topology • Software requirements • Availability	• Security (+) • Client Requirements • Multiple hosting • Application analysis • Web directories • Network topology(+) • Software requirements • Availability

continued

Models	*Intranet*	*Extranet*	*Internet*
Distributed	• Security • Multiple servers • Application analysis • Web directories • Network topology • Software requirements(+) • Availability(-)	• Security (+) • Client Requirements • Multiple server • Multiple hosting • Application analysis • Web directories • Network topology(+) • Software requirements(+) • Availability	• Security (+) • Client Requirements • Multiple servers • Multiple hosting • Application analysis(+) • Web directories • Network topology(+) • Software requirements(+) • Availability
Enterprise Distributed	• Security • Multiple servers • Application analysis • Web directories • Network topology(+) • Software requirements(+) • Availability(-)	• Security (+) • Client Requirements • Multiple servers(+) • Multiple hosting • Application analysis(+) • Web directories • Network topology(+) • Software requirements(+) • Availability(+)	• Security (+) • Client Requirements • Multiple servers(+) • Multiple hosting • Application analysis(+) • Web directories • Network topology(+) • Software requirements(+) • Availability(+)

Chapter 3
Security

This chapter focuses on the functional implementation of the Domino Go Webserver security services as outlined Chapter 1: Architecture.

- Access Control
- Identification and Authentication
- Confidentiality
- Data Integrity
- Non-Repudiation

Secure Sockets Layer

SSL (Secure Sockets Layer), developed by Netscape Communications Corporation, is a security protocol which encrypts data sent by an application using the TCP/IP sockets interface. The intent of SSL is to ensure private and authenticated communications. SSL is an open standard that Netscape has placed into the public domain. “The SSL Protocol Version 3” Internet draft can be obtained from:

`http://home.netscape.com/eng/ssl3/draft302.txt`

SSL allows clients and servers to communicate confidential data through the Internet, extranet or intranet. HTTP is one such application that benefits from SSL. The Domino Go Webserver offers this nonproprietary technology SSL Version 3.

In this section, SSL and how to use SSL for secure communications is described.

SSL Overview

SSL is based on cryptography, and can be used to ensure the privacy and the integrity of data traveling over a non-secure network. In addition, SSL enables the receiver of data to identify and authenticate the identity of the sender. This is an important concept, since transactions conducted over the Internet will often be between two parties that have not established their identities.

Detailed information about SSL protocol is available from:

`http://home.netscape.com/newsref/std`

To use SSL, both the client and server need to support this protocol. The SSL protocol delivers the necessary functionality to implement certain security services such as:

- Authentication
- Confidentiality
- Data Integrity

As information travels over the Internet it can be routed through many different intermediary computer systems. Each of these intermediary systems represents a potential security risk. SSL enables the encrypted transmission of data over the Internet which otherwise would be sent in the clear (unencrypted) and become vulnerable. SSL uses authentication and encryption technology developed by RSA Data Security, Inc.

The United States Government imposes restrictions on the size of the encryption key that may be used in software exported outside the U.S. These rules are currently under review, but the present effect is to limit the key size to 40 bits. The International (export) versions of software products have this restrictive security built into them. SSL handles mismatches between the export and non-export versions in the negotiation phase of the handshake. For example, if a U.S. browser tries to connect with SSL to an export server, they will agree on export-level-strength encryption.

According to Netscape, a message encrypted with a 40 bit key takes an average of 64 MIPS years to break (meaning a 64 MIPS computer would need a year of dedicated processor time). The U.S. version uses a 128 bit key which would require an enormous amount of computing resource to break. Both of these key sizes use RC4 stream encryption technology. RC4 is referred to as symmetric key encryption algorithm. This type of encryption uses a shared secret key and is more efficient for encrypting bulk (streaming) amounts of data. RC4 is a proprietary bulk cipher from RSA Security, Inc. See Appendix C for more information regarding cryptographic techniques. Sever authentication uses RSA public key cryptography with ISO X.509 certificates.

Key pairs or a public/private key encryption scheme is based on all information being encrypted by the sender with the public key of the receiver. The receiver decrypts the information with their private key. Therefore, anyone can send an encrypted message by using public information but the message can only be decrypted by the receiver's private key which is only accessible by the receiver. The only requirement is that public keys be associated with their users in a trusted manner, for example in a directory. Public key encryption can be used for both confidentiality and authentication (digital signatures). Digital signatures provide a method to determine if the contents of a message have been altered (data integrity) as well as to verify the identity of the signer (non-repudiation).

SSL Protocol

The SSL protocol is composed of two layers or sub-protocols. The lowest layer, which is layered on top of the transport protocol TCP, is the SSL Record protocol. The SSL Record protocol allows the server and client to authenticate each other and negotiate encryption keys before the application transmits any information. SSL is application protocol independent. This means that other application types such as FTP, Telnet, SMTP, and NNTP can benefit from SSL technology.

SSL is composed of two sub-protocols:

- **SSL Handshake Protocol**

  This initializes a secure session in which the session partners introduce themselves and negotiate session characteristics, by authenticating the server (and optionally, the client), agreement of encryption scheme, and transfer of encryption keys. This handshake is performed using asymmetric encryption scheme (public - private encryption key pairs).

- **SSL Record Protocol**

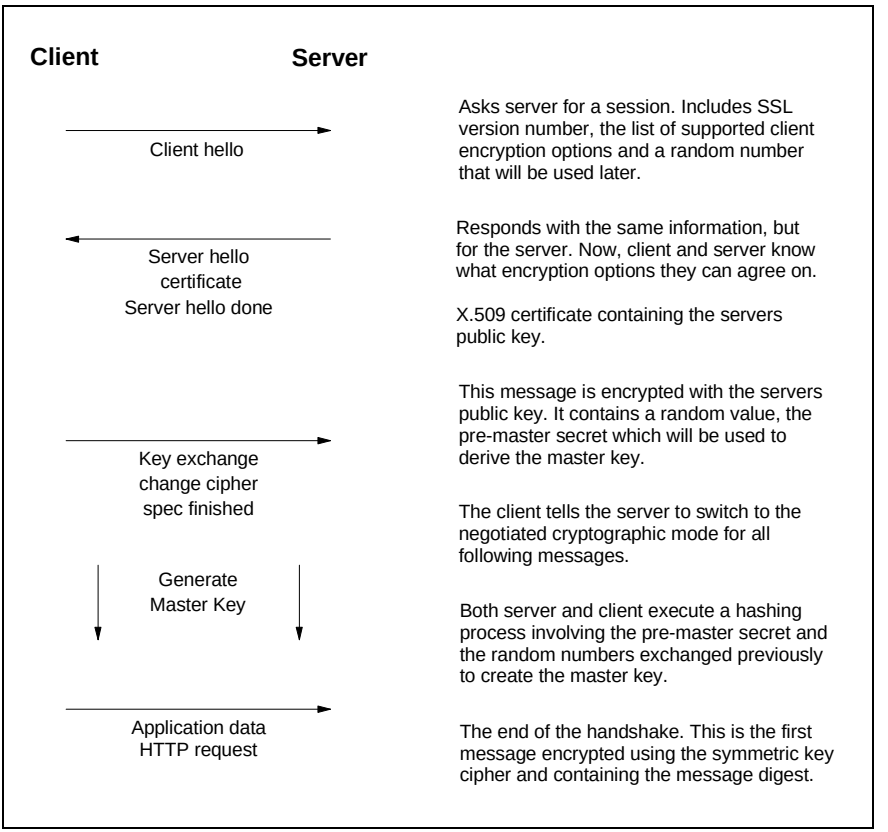
  This transfer application data using the encryption scheme and keys agreed upon during the handshake phase. Usually, this phase is performed using a symmetric encryption key scheme, since such schemes provide better performance than asymmetric schemes. The protocol allows a number of different schemes to be used, including symmetric schemes such as Data Encryption Standard (DES), hashing functions, or various asymmetric schemes.

Establishing Secure Server Communications with SSL Handshake

In the handshake part of the SSL protocol, the client and the server communicate with each other to set up a secure communication and encryption parameters (such as type of encryption keys).

The client and server exchange *hello* messages. The hello messages contain information about the capabilities of the client and server, for example a list of cipher suites (encryption keys), combinations of cryptographic algorithms and key sizes that will be accepted for the session. The server provides a pubic key certificate. This is the technique by which SSL checks identity and authenticates the session partner.

The example below only shows server authentication, but if client authentication were required there would be another message exchange using the client public key.



The SSL protocol uses both public key (asymmetric) and symmetric key cryptography. Public key encryption is used for the handshake. During the handshake the master is passed from client to server. The client and server make their own keys using the master key. The session keys are used to encrypt and decrypt data for the remainder of the session.

Client Authentication

Domino Go Webserver supports client authentication through the implementation of SSL version 3. Client authentication is an additional step in the SSL handshake protocol. The server can request a certificate from the client when it sends its own certificate. The flow of the SSL handshake with client authentication is summarized below:

1. The server responds to the client-hello request with the message including its own certificate, and a request for a client certificate.

2. When the client receives this message, and there is a suitable certificate, the client sends this certificate to the server. If the client does not have a certificate, a no certificate alert is sent to the server. In this case, the server will decide whether to proceed with communication based on its configuration.

Client authentication is enabled by changing the "httpd.cnf" configuration file directive:

SSLClientAuth

After the client authentication process, the server can control the access of resources by using the client certificate information instead of by user id and password protection. This function is enabled by coding SSL authentication parameters on protection setups, ACLs files or both. For example, you can change the protection directive for user id "Mike" as shown below:

With userid and password:

Protection USERID_AUTH {

AuthType *Basic*

Passwd File *%%SAF%%*

Mask *Mike*

}

With certification information:

Protection SSL_CL AUTH {

CommonName *Mike*

Country *US*

Organization *"Sample Client Certificate"*

Mask *Anybody@(*)*

}

Client Authentication Setup

When using the client authentication function, you need to perform the following steps:

1. Server side. Change the *SSLClientAuth* directive in the "httpd.cnf" file to *ON*.
2. Client side. You must have a certificate for the client and you must specify it on the security configuration panel of the browser as a personal certificate.

When using an external CA (Certificate Authority) organization, the browser and Domino Go Webserver must both have the certificate of the CA who certified the client and set it as a trusted root.

When using your Domino Go Webserver as the CA, use Domino Go Webserver to create the client certificate. In this case, you should send back the client certificate with a CA certificate from Domino Go Webserver. The client will then receive the CA certificate and designate it as a trusted root.

For example, when using Netscape Navigator, you can set up the personal certificate from the security preferences folder of the option menu. After setup, the personal certificate panel will appear as shown below:

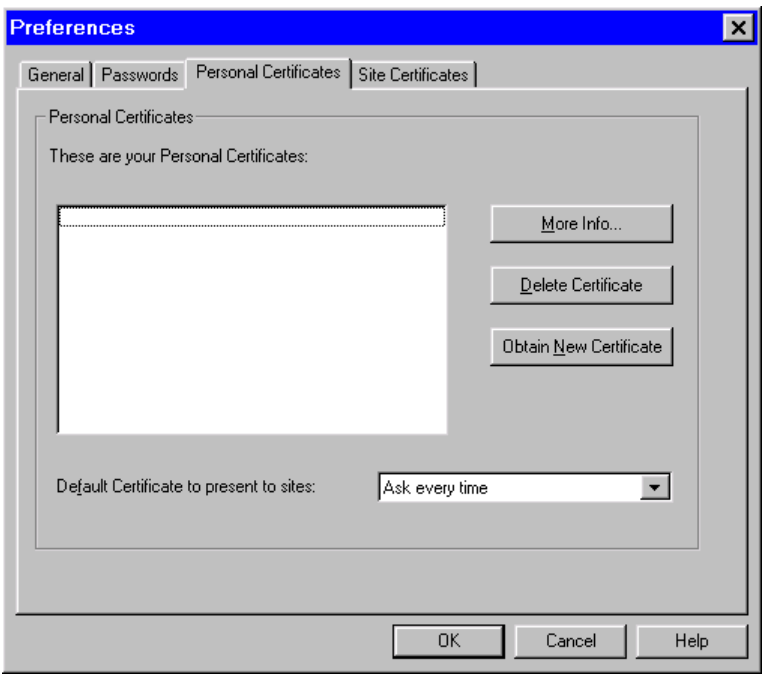


Figure 3.1. Netscape Personal Certificate Panel

Note In this example, the option “Default Certificate to present to sites” is set to “Ask every time.” At this option, Netscape asks you which certificate you want to use whenever requesting to the server. To avoid this, specify the actual certificate as a default.

3. To protect the resources based on the certificate, specify the appropriate directives for protection in the “httpd.cnf” file, or make ACL files which define the access control with the certificate information.

How to Identify a Secure SSL Connection

The negotiation and authentication process of the SSL handshake is rather complex, but fortunately it is transparent to the user. The way to determine if the connection is secure is by looking at the URL field. If the URL field begins with **https://,** then the connection is secure. To start a secure connection with a server that provides security using the SSL protocol, insert the letter "s" so the URL begins **https://**.

Once the SSL connection has been established the browser gives the user a visual indication. In the case of Netscape Navigator this is a key symbol at the lower left of the screen as shown below:

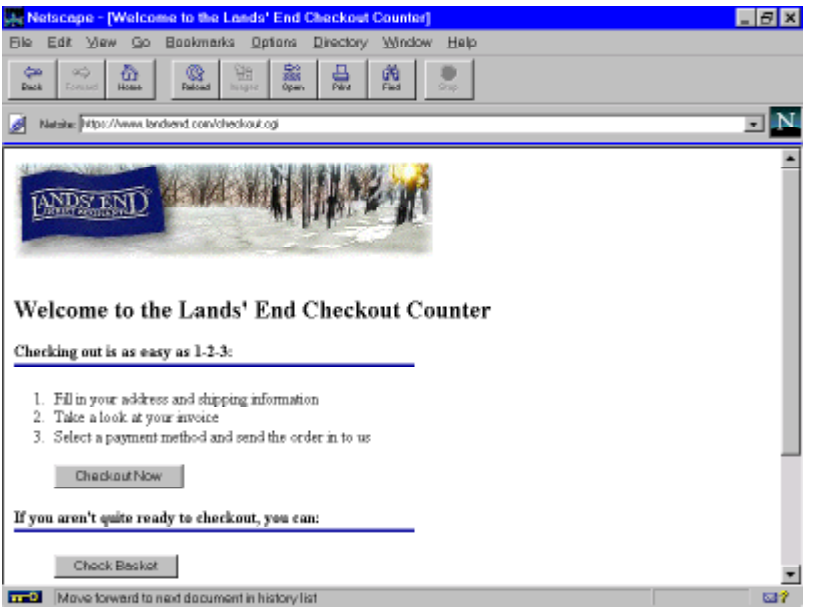


Figure 3.2. Secure SSL Connection Displaying Key Symbol

From the point of view of the Webmaster, SSL is also quite simple. First the Webmaster needs to generate a key pair for the server, and obtain a certificate for it. Normally this involves providing documentation to a certifying authority and paying an annual fee, although it is possible to generate your own certificates for testing and intranet use.

Once the server certificate has been installed, the Webmaster can create HTML links with the "https:" prefix to cause SSL to be invoked. For example:

```
<A HREF=https://my_server/secret.doc>Go into SSL</A>
```

What Is Authentication?

Authentication is the process used to verify identity. There are two ways in which the server uses authentication:

- Digital signatures
- Digital certificates

A digital signature is created by running a message through a hashing algorithm. The result of this process is known as a message digest. The message digest is then encrypted with the sender's private key. The receiver of this particular message can only decrypt it with the public key of the sender. Since the private key is possessed only by the sender, the digital signature provides a way to identify its source with certainty (non-repudiation). Once the message has been decrypted, then the message digest is recalculated. The two message digest values are compared, if the values match, then the message has not been tampered with (data integrity).

Note The RSA public key encryption algorithm is two way. So a message can be encrypted with either the public or private key and be decrypted with the inverse key.

What Is a Certificate?

A certificate is the electronic form of proof of identity. When you conduct transactions in person (for example, when opening a bank account), you are required to prove your identity. You might do this with documents that are trusted by the bank, such as a passport or identification card, issued by a government. In order to conduct electronic transactions, both parties to the transactions need to prove their identities. Since it would be impractical to meet to show passports, the idea of a digital proof of identity has been invented. These "digital certificates" are issued to people or organizations that request them, by a trusted third party. These trusted third parties are called certification authorities (CA), and have the same responsibilities as organizations that issue passports, driver's licenses, or identification cards.

A certificate is made up of:

- The public key of the person being certified.
- The name and address of the person being certified, also known as the *Distinguished Name.*
- The digital signature of the CA.
- The issue date.
- The expiration date.

SSL and Certifying Authorities

Authentication in SSL depends on the client being able to trust the server's public key certificate. A list of top-level authorities is pre-installed, for example with Netscape Navigator. The illustration below shows part of the list of CAs.

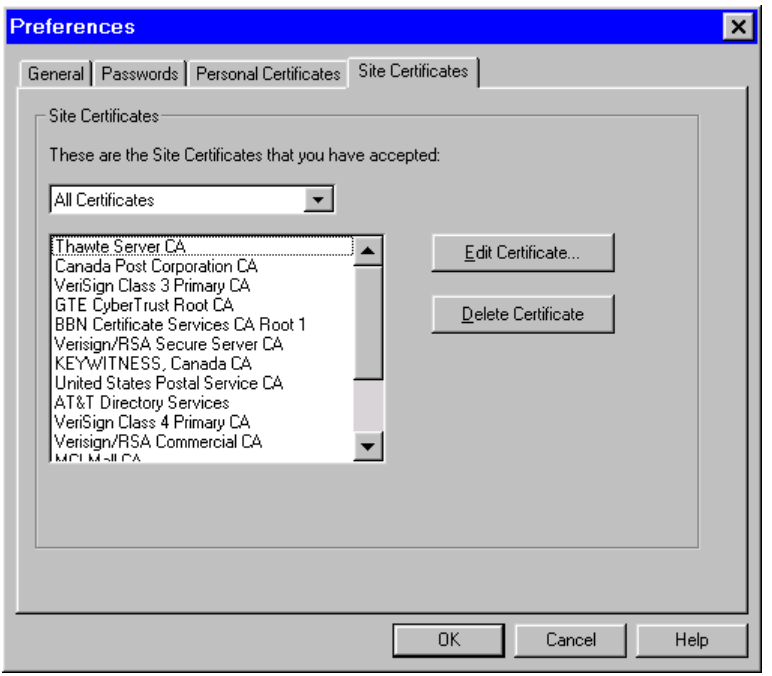


Figure 3.3. Netscape Site Certificates Panel

A certificate links the description of the owner of a key pair to the public part of the key. The validity of a certificate is guaranteed by the fact that it's signed by some trusted third party, the certifying authority (CA). But how does a certifying authority become trusted? In the case of an SSL-capable browser, the certificates of trusted authorities are kept in a key database, sometimes called a key ring file.

This approach has the benefit of being very simple to set up; a browser can authenticate any server that obtains a public key certificate from one of the CAs in the list, without any configuration or communication with the CA required. However, there are also some problems arising from this method. The first is that a CA will not automatically be recognized until the browser has been updated. The second problem is that there is no way for certificate revocations to be processed. (For example, if a CA determines that a public key owner is fraudulent after a certificate is issued, the certificate will remain useable until it expires, without the end user being aware of any concern.)

The browser vendors have a two-part scheme to overcome the first problem (new CAs):

1. There is a special MIME format, which allows a browser to receive a new CA certificate that has been signed by one of the known CAs. However, this method is not widely implemented yet.

   See **`http://home.netscape.com/eng/security/download.html`** for details of download formats.

2. The browsers will tell you that you are connecting to a secure server whose certificate is not from a known CA. You can then elect to trust just that server (that is not the CA that signed the server's certificate).

This latter approach is useful in an intranet environment, where you may want to create an enterprise only CA. For Internet applications you should purchase a certificate from one of the known CAs, such as VeriSign. For example, the following graphic is the VeriSign Web page from which a server certificate (server id) can be purchased:

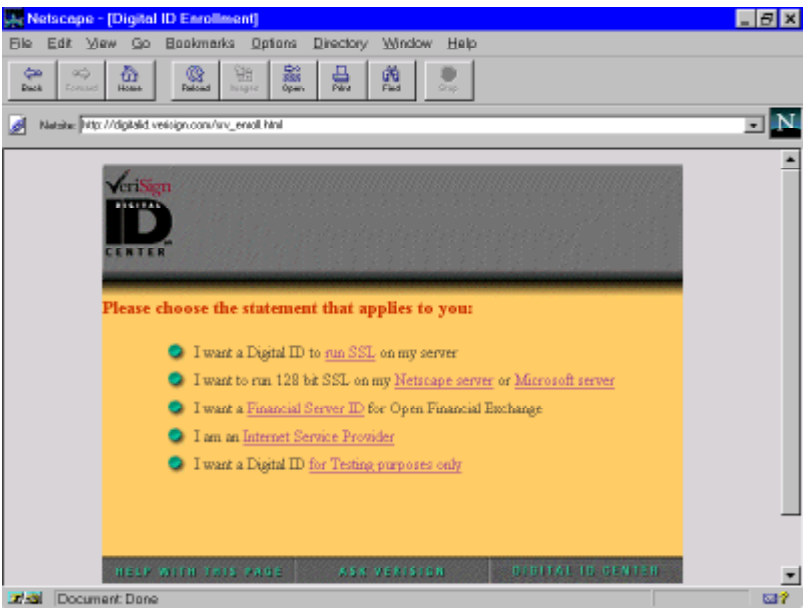


Figure 3.4. VeriSign Web page

Obtaining a public key certificate from a CA provides proof of identity and it is reasonable to assume that the level of proof needed for a client is much lower than that needed for a server. Before providing a server certificate, the CA will require documentary proof of the legitimacy of the request. In the client case, this proof can often be provided online, because a lower level of

checking is needed. This is especially true of an intranet environment and certificate server products are initially intended for organizations that want to set up an internal authentication process.

SSL authentication relies on the session partners recognizing the CA that signed a certificate as a trusted entity. There are limitations in the way this is currently implemented. The challenge is to create a scheme in which CAs can exchange information with other CAs, client and server systems. This information includes information about new CAs and revoked certificates.

Managing Keys and Certificates with Domino Go Webserver

Domino Go Webserver uses RSA public key cryptography for encryption, message digest and authentication. Domino Go Webserver provides forms that can be accessed by a browser to manage your keys, key ring file passwords, and certificates.

Domino Go Webserver can act as a CA, that is you can sign your own or anyone else's certificate request. This is an appropriate choice for implementing Domino Go Webserver within an intranet that has no outside Internet connectivity.

The following graphic is the "Configuration and Administration Forms" document from the Domino Go Webserver displaying the security section.

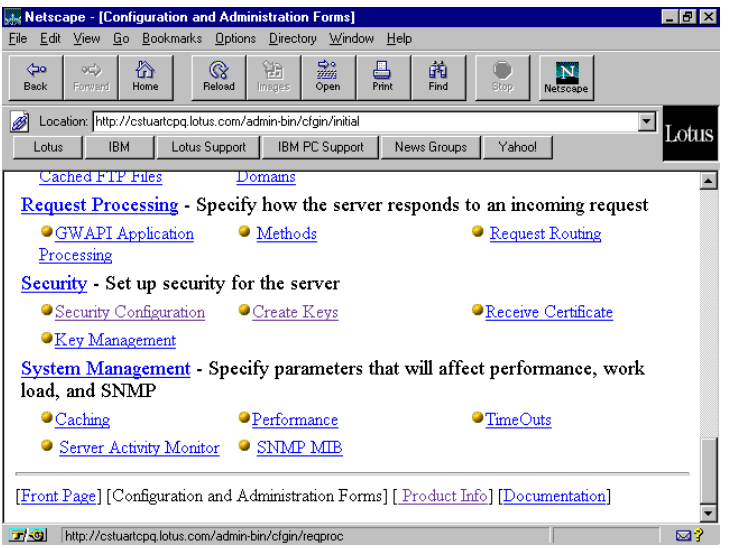


Figure 3.5. Configuration and Administration Form

Domino Go Webserver configuration can be accomplished either by directly editing the “httpd.cnf” configuration file or through the “Configuration and Administration” document. The following graphic is the SSL and key ring form.

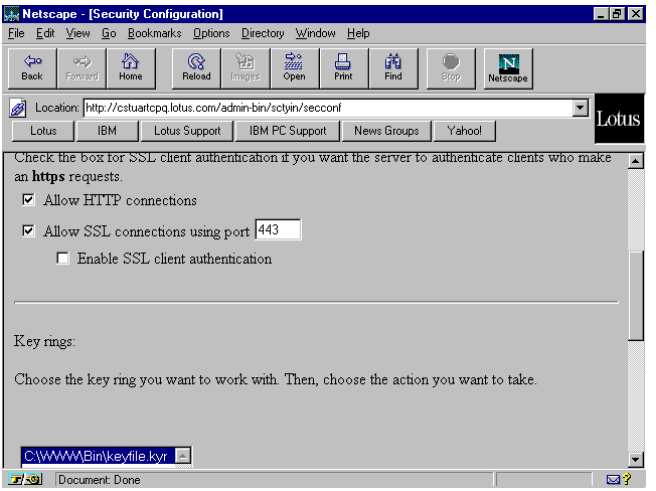


Figure 3.6. The SSL and Key ring form

Notice in the above figure that SSL client authentication could be enabled through this form, as well as specifying an SSL port.

Also, a request to obtain a certificate from a CA could be processed through a Domino Go Webserver security from that produces a file which then needs to be mailed to the CA, as shown in the following figure.

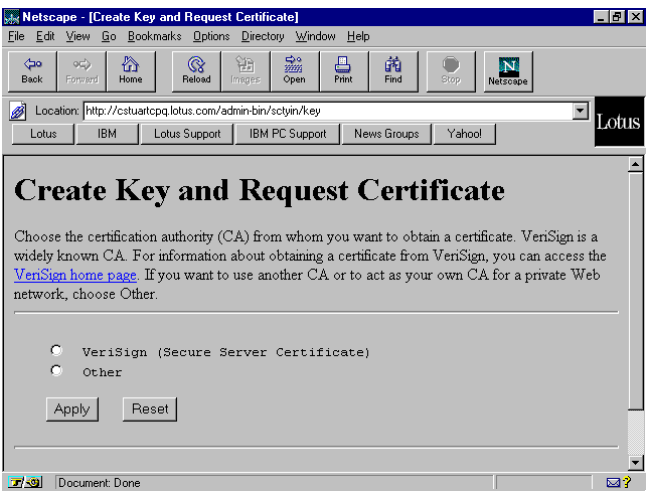


Figure 3.7. The Create Key and Rsquest Certificate Form

Value of SSL Support for Domino Go

Since the Internet is so easy to access, and is a routed network (as opposed to point to point), it raises security concerns when used as the infrastructure for any sort of electronic communication. The Internet should be considered a non-secure network, and you should assume that any data you send on it can be read by any person. In a similar way, corporate intranets based on TCPI/IP should be considered less secure than traditional private networks, such as those based on SNA.

SSL is one of the major security functions designed for these new networks for authentication of users (through certificates) and data integrity and privacy (through cryptographic technology). Support for the latest level of SSL in Domino Go Webserver means your server can participate in these secure conversations.

SSL Tunneling

Domino Go Webserver runs not only as a Web server, but also as a proxy server that enables users to access servers that exist outside of their network without showing where they are coming from. SSL Tunneling is an additional proxy server function that allows users to access the external secure server using SSL protocol through a proxy server.

SSL Tunneling Function Overview

In a normal proxy functional environment, when accessing the remote server by SSL protocol, SSL function is not needed in the browsers, but is needed in the proxy server. The connection between the client and the server is shown below.

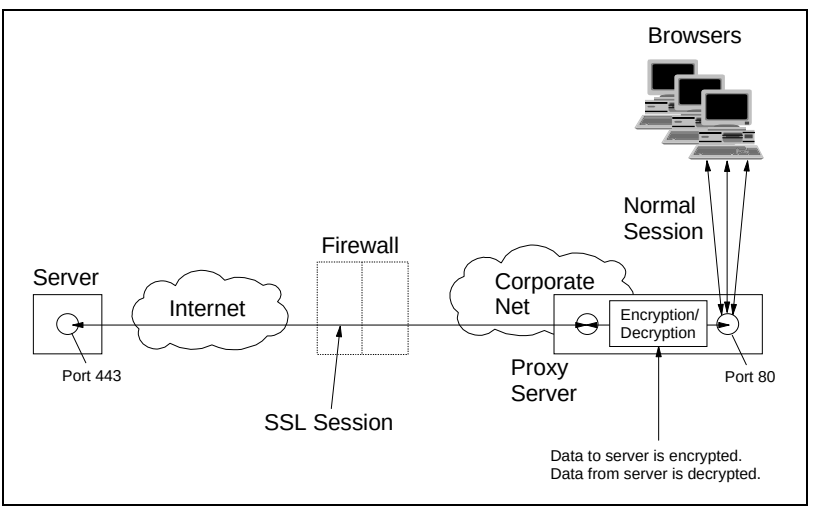


Figure 3.8. SSL Connection with Proxy Server

The proxy server is responsible for all functions of the SSL protocol. Once the request is received, the proxy server establishes an SSL session with the remote server and sends data to the client. However, this approach has disadvantages:

- The connection between the client and the proxy server is normal HTTP and therefore is not secure.
- The proxy server needs to have full SSL implementation.
- For the server side, only the proxy server is recognized as the client. Therefore, the server cannot know who the actual requester is, even if the server is using the client authentication of SSL.

SSL tunneling is a generic protocol for transporting an application-level protocol (like http-over-SSL) through a proxy server, without requiring that the proxy server speak that protocol, and without giving the proxy server access to sensitive data. In this case, the connection is changed to that shown in the following figure.

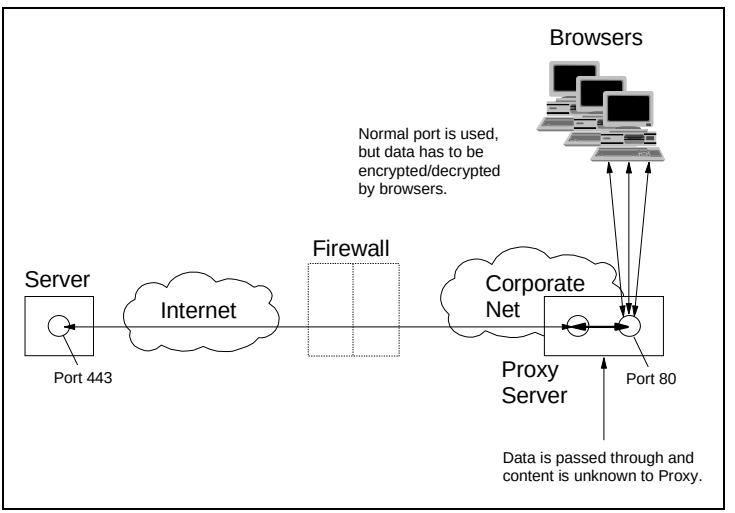


Figure 3.9. Proxy Server with SSL Tunneling

Basically, the client asks the proxy server to connect to the destination server by using a URL-specified *https:*. But the proxy server just passes data to the port defined as the destination, without interpreting it. Therefore, the client's browser must do all the secured function in this environment. Using SSL tunneling, the proxy server does not need to become a secure server. However, you have to use a browser which supports SSL and the SSL tunneling functions.

Setting Up SSL Tunneling in Domino Go Webserver

The procedure to set up SSL tunneling in Domino Go Webserver is described in *Domino Go Webserver Webmaster's Guide*. You first have to set up Domino Go Webserver as a proxy server, and then add some SSL tunneling specific definitions. In summary, the steps to configure SSL tunnel are as follows:

1. Enable the CONNECT method in the "httpd.cnf" file.
2. Specify the Pass rule to the destination server in the "httpd.cnf" file.

   Pass *:443

3. Set the security proxy in the browser.

For example, in the case of Netscape Navigator, you should specify the server name and normal port (not the SSL port) on the security proxy box in the proxies folder. A sample panel is shown below:

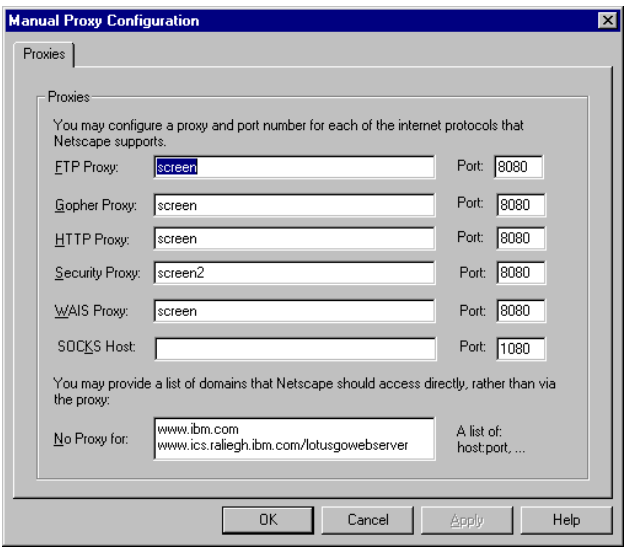


Figure 3.10. Netscape Proxies Panel

Value of SSL Tunneling for Domino Go

As we mentioned before, there are some advantages in having the proxy server maintain the SSL protocol, instead of having the client handle it. SSL tunneling will remove these disadvantages. It is also especially valuable for the application server to be able to request a certificate from the actual client.

In addition, SSL tunneling is used not only for the SSL protocol, but also for other security functions that use a specific port number. Because the server using SSL tunneling does not recognize the content and the format of data, it does not mention the encryption scheme, which enhances system security.

SSL Tunneling Considerations

SSL tunneling is generic, but being able to use this function in your proxy server will depend upon whether the client supports it or not. For example, Netscape Navigator uses tunneling to establish a secure connection through a proxy server.

SSL tunneling does not use a caching proxy function, so there may be performance and network capacity impacts associated with these transactions.

SSL tunneling does not depend on a secure internal network. It encrypts and decrypts at the client. So, the internal network can be unsecured without causing a security hole.

Domino Go Webserver as a Proxy Server

Access to Web sites over an unsecured public network that originates the request from a client connected to an internal corporate network benefit from a proxy server because it permits client anonymity. Since the proxy server is submitting the request on behalf of the client, it prohibits the server from obtaining what could be considered by company security policy private information (for example a TCP/IP address). Client information could potentially be used to construct mailing lists or identify Web site search patterns and this information could possibly be resold to other organizations.

Another, proxy server benefit is caching frequently accessed Web pages. The caching of Web pages minimizes the actual number of Web page retrievals made by the client. Reducing the number of client Web page retrievals has the potential to improve user performance.

Many Web browsers also support the local caching of documents, so a part of the planning is to decide where and when to cache Web page documents. Some issues to consider are as follows:

- Cache location (server, client or both)
- Cache size
- Which documents to cache
- How long to keep the cache documents

The Domino Go Webserver “Configuration and Administration” document has several proxy settings forms as listed below.

1. Proxy Server Setting
2. Cache Settings
3. Caching Filters
4. Cached File Expiration

5. Time Limit Cached Files
6. Time Limit for Unused Cached HTTP Files
7. Expiration Settings for Cached FTP Files
8. Proxy Chaining and Non-Proxy Domains
9. Cached Storage Reused

The following figure shows the “Configuration and Administration” form with all the security settings for Web page forms.

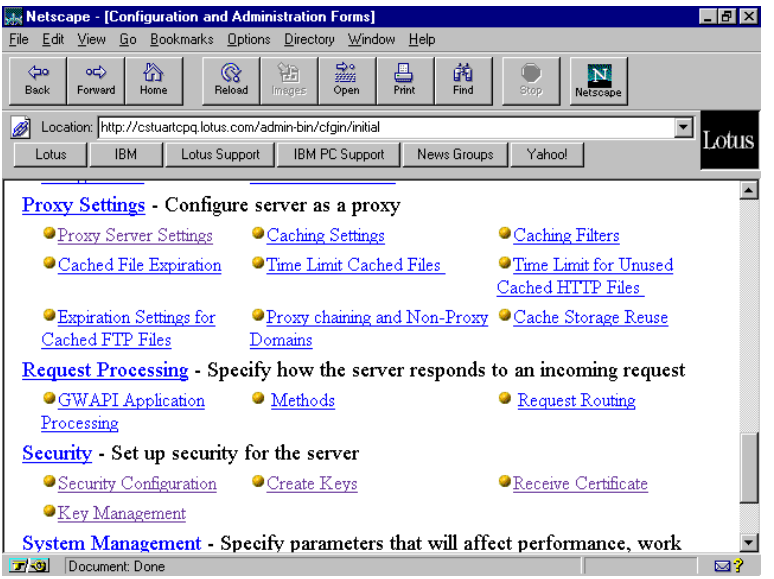


Figure 3.11. The Configuration and Administration Security Forms

The cache settings can be changed by directly editing the “httpd.cnf” configuration file, for example the cache size directive coded in the configuration file is:

CacheSize 50Mb

Through the cache settings form the cache size can also be changed. The cache size is the maximum amount of disk space that can be used by the proxy cache. The default cache size is 5Mb. When the size of the proxy buffer cache needs to grow larger than what is allocated, the garbage collection process begins. The server garbage collection process deletes files that should no longer be cached. Files are deleted based on their expiration date and other proxy directive statements. The following two figures are the Web page forms for submitting a cache size change and the subsequent confirmation of the change.

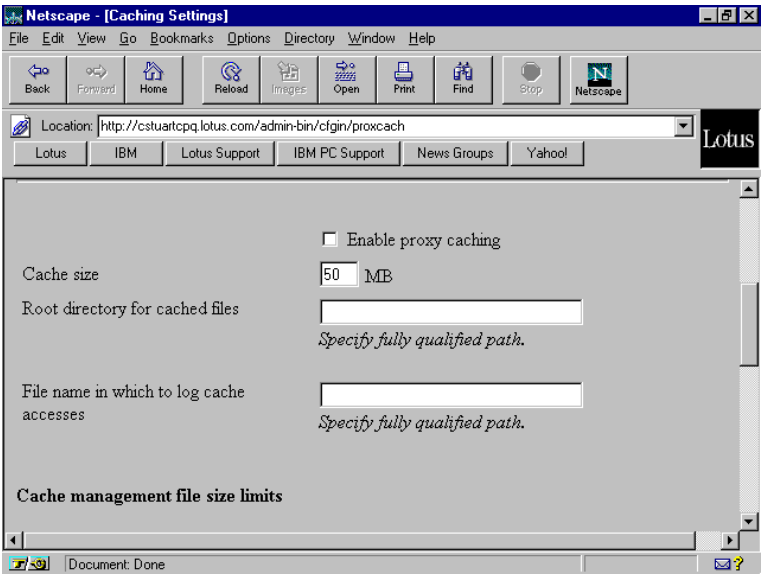


Figure 3.12. The Cache Settings Form

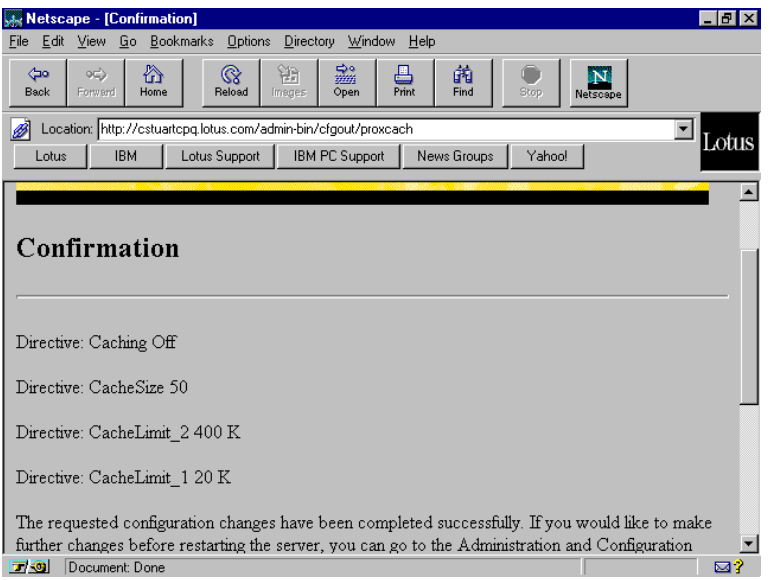


Figure 3.13. The Cache Change Confirmation Form

Through the proxy server settings form the proxy buffer size can be changed. The proxy buffer is the buffer into which the proxy server reads files before sending them to clients. This Web page form also permits the selection of different protocols that the proxy server can handle.

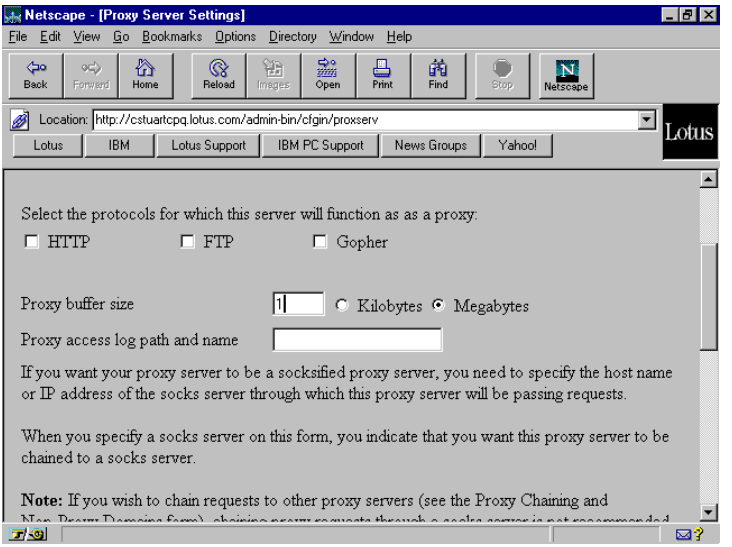


Figure 3.14. The Proxy Server Form

Another proxy server consideration is caching filters. Caching filters determine which URL's documents will be retrieved by the proxy server and cached locally. If no caching filters are specified, then requests for URLs will be cached. The following figure is an example of specifying URLs as a part of a cache filter.

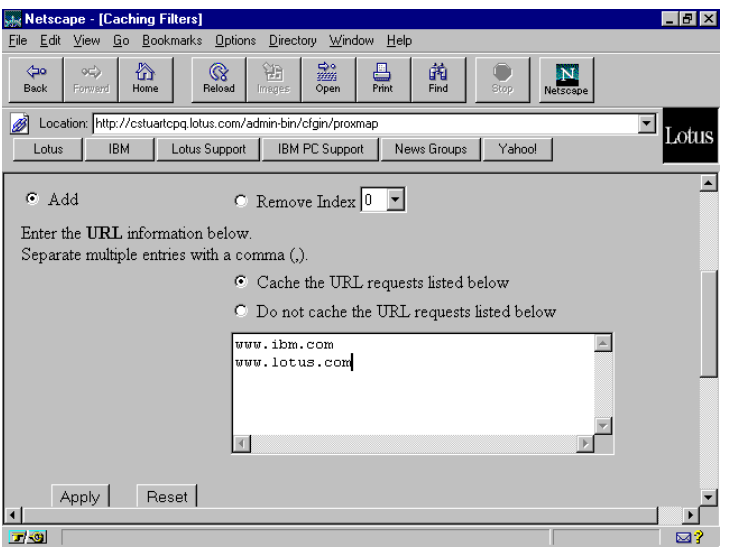


Figure 3.15. The Cache Filters Form

For a detailed description of all the Domino Go Webserver proxy server settings, refer to the *Webmaster's Guide.*

Access Control and Document Protection

Domino Go Webserver uses a concept known as protection setups to control access to the server resources. A protection setup is a group of protection subdirectives that work together to define how the server should control access to the resources being protected. Protection setups can be defined within the "httpd.cnf" configuration file, in separate protection setup files, or by using the Administration and Configuration forms.

Protection Setup Methods

Protection setups are linked to files based on the requests that are used to access those files. There are two directives used to define the requests you want to protect:

- *DefProt*
- *Protect*

Use the *DefProt* directive to associate a default protection setup with requests that match a template. The *Protect* directive is used to activate protection setup rules for requests that match a template. A protection setup can also be defined as part of the DefProt or Protect directive.

The example that follows uses a separate file to contain the protection subdirectives.

```
DefProt    /secret/*  /server/protect.setup1.acc
```

Another example;

```
DefProt    /secret/*  /server/protect/setup1.acc
Protect          /secret/scoop/*
```

In the above example, requests that start with /secret/scoop activate protection. The protection setup is defined in the setup1.acc protection setup file.

Note For protection to work properly, you must put your *DefProt* and *Protect* directives before any Pass or Exec directives in the configuration file.

For a detailed description of the *DefProt* and *Protect* directives refer to the *Domino Go Webserver Webmaster's Guide.*

Protection Subdirectives

There are several protection subdirectives that can be coded as part of the protection setup. For example:

- **AuthType** - Specify the authentication type.
- **DeleteMask** - Specify the user names, groups, and addresses allowed to delete files.
- **PasswdFile** - Specify the location of the associated password file.

- **PutMask** - Specify the user names, groups, and addresses allowed to put files.
- **ServerID** - Specify a name to associate with the password file.

For a detailed description of the protection subdirectives refer to the *Domino Go Webserver Webmaster's Guide.*

Access Control Lists

To limit access to specific files on a protected directory, create an Access Control List (ACL) file and place it on the directory. The ACL file must be named .www_acl. The mask subdirectives in the protection setup usually define the first level of access control and then the ACL file further limits access. However, if you want all control to come from the ACL file, use the ACLOverride subdirective in the protection setup. This causes the mask subdirectives in the protection setup to be ignored. All access is passed to the ACL file.

ACL files can be coded within the "httpd.cnf" configuration file or through the Administration and Configuration forms, as the following figure shows.

Figure 3.16. The Access Control Form

For a detailed description of Access Control Lists, refer to the *Domino Go Webserver Webmaster's Guide.*

Platform for Internet Connection Selection (PICS)

In order to support the emerging emphasis on rating Internet content, Domino Go Webserver has added support for the Platform for Internet Content Selection (PICS). This section describes how to go through the rating process for your site and use the Domino Go Webserver to run a PICS rated site on the Internet.

The Internet and Its Content

The Internet has proven to be a convenient, inexpensive way to communicate, advertise, and disseminate your information. It has provided a means to tell the world your message, whether it be about a service, product, or an idea. As evidenced by the growing numbers of Web sites and the amount of advertising done on the WWW, we have seen that many people have taken advantage of this medium to communicate their message. The problem is, the Internet medium is no different than any other. The message desired by some people can be found very offensive by others.

Most people would agree that it is unacceptable to allow unsupervised children to view pornographic pictures and read hate speech. However, there may be more disagreement on acceptability when the subject material is not as obvious. For instance, there may be certain contexts whereby the content is acceptable to some people and not others. For example, suppose a science-related site on the WWW, set up by an educational institution, contains content on human reproduction. This site may be educational to some and offensive to others, so defining and solving the problem becomes much more complex.

Software manufacturers have recognized the problem (or capitalized on the opportunity) and produced a variety of filtering products and utilities to attempt to block Web content from the browser and restrict access to sites based on their content. In most cases, vendors have taken it many steps further and not limited their products to only blocking Web site content. They also filter search engine strings, returning no hits for “bad” searches. Some prevent the dissemination of personal information by replacing credit card numbers, names, addresses and phone numbers, as defined by a list, with a string of X’s. That way the intended recipient receives useless data. Most filtering products log online sites visited, time spent, and searches attempted, among other details. If this is what you want, you can choose from a variety of products.

Although they sound like a solution, Internet filtering utilities have some inadequacies associated with them. These products are based on a string comparison routine against lists that are stored in a database. Although editable and updatable with periodic refreshes, the lists are never complete. New sites enter the WWW daily that contain material deemed objectionable

by the very user of these products. Attempting to maintain the databases with these refreshes proves to be a formidable and never-ending task. Filtering utilities may help lessen the problem, but they are not a complete solution in and of themselves.

There needs to be a methodology whereby the end user can ensure censorship based on individual criteria easily and permanently, while protecting freedom of speech for everyone else. Paul Resnick, one of the creators of PICS, sums up the issue by saying:

"It's time to go beyond the freedom of speech versus protection of children debate. The larger issue is sharing: how can a diverse user population share a single communications medium without constantly being at each other's mercy? If we insist on the Net as a global free-for-all, some people will opt out and others will try to impose their notion of order on everyone else."

PICS may be the vehicle by which the Internet can be shared.

What Is PICS?

The Platform for Internet Content Selection is a standard for using metatags to provide information about information. These metatags, which are stored with or sent with each HTML page, rate the content of the information contained in each HTML page. The content rating can be a set of categories pertaining to levels of nudity, offensive language, violence, and sexual content, for example. Some rating systems rate Web content using a scale of values for categories such as educational value, questionable/illegal, and even "coolness." Every day we are exposed to content ratings, whether it be for movies, television, or musical recordings. PICS is a methodology to rate Internet content.

Who Can Rate Web Sites?

Web sites can rate themselves through a PICS self-rating voluntary service, or be rated by a third party, called an independent rating service.

Self-Rating Voluntary Service

To use a self-rating voluntary service, you connect to its Web site and describe your Web content by filling out a questionnaire. After completing the questionnaire, the service gives you a text label in a special format that can be embedded in your HTML pages or added to a Domino Go Webserver label file.

The World Wide Web Consortium publishes a list of both self-rating and third-party rating services. This list is located at:

- World Wide Web Consortium - (**`www.w3.org/pics`**)

The following are examples of some of the self-rating voluntary service bureaus:

- RSACi - (**www.rsac.org**)
- Safe For Kids - (**www.weburbia.com/safe**)
- SafeSurf - (**www.safesurf.com**)

The most popular of these three is the Recreational Software Advisory Council on the Internet (RSACi). This organization has rated over 30,000 Web sites and welcomes you to rate your site using RSAC self-rating system. This rating system is designed to rate Web sites based upon levels of nudity, language, violence, and sexual content. The following figure shows the home page of this Web site.

Figure 3.17. The Recreational Software Advisory Council Web Page

Independent Rating Services

Independent rating services rate materials published by others. As a Web site author, you cannot control the rating these services will assign you. Some of the more common independent rating services are:

- CyberPatrol's CyberNOT list - (**www.microsys.com/pics/pics_msi.htm**)
- NetShepherd - (**www.netshepherd.com**)

The labels produced by the independent rating services are not usually sent to be imbedded in HTML pages, but distributed via label bureaus.

Why Rate Your Web Content?

Having your Web site rated is something that you may want to consider if you want your site to be viewed at all by PICS-enabled browsers. As proliferation of PICS becomes a reality, you may find it necessary to rate your site to make it accessible to the widest possible audience. Also, the federal government is considering possible legislation to rate Web content. Certainly, in order to be a good "netizen" (Internet citizen) you could at least self-rate your site as a whole by using the RSAC rating system.

PICS-enabled clients allow users to determine which rating services they want to use and, for each rating service, which ratings are acceptable and which are unacceptable.

For example, a family might choose a rating service that rates documents according to their sexual content. The rating service might have a low rating for romance, a higher rating for passionate kissing, and yet higher ratings for more explicit sexual activity. The parents might decide that documents containing romance are the highest acceptable rating for the household. They would then configure their browser to reject all documents that are unrated or contain a higher rating from this rating service.

Another example, is the ABC Corporation could label its own documents with a "For ABC Use Only" and could equip all its employees with browsers configured to accept only documents with that rating.

There are several steps in this process:

1. The client sends a request.

   When a PICS-enabled client requests a document, it indicates in the request which rating services are of interest when rating the document content.

2. The server sends a response.

   Assume the PICS-enabled server has a copy of the labels the client is requesting. When the server receives the client's request, it sends the labels along with the requested document However, if the server does not support PICS or does not have copies of labels from that particular rating service, it sends the requested document anyway.

3. The client checks the server response first.

   The client first checks to see if the requested ratings labels are embedded in the document (in the meta information) or if they were sent along with the document. Some clients might accept rating information that is embedded in the file. Others might require a separate label from a registered rating service and a guarantee that it was created by that service. If the client actually finds the label information it wanted, it evaluates the rating and either displays the document or blocks it and displays a message.

4. The client contacts the rating service, if necessary.

   If the client does not receive the label information with the requested document from the server, it might send a subsequent request directly to the rating service asking for the label information for that document.

Different Ways of Enabling PICS

You can either manually embed the rating information label that rates your site's content in your HTML pages, or you can use a function that is part of Domino Go Webserver to store the rating labels in a central file. Storing them in a central file under the control of the Domino Go Webserver makes the administration of your rating labels easier. For a detailed description of PICS implementation refer to the *Domino Go Webserver Webmaster's Guide.*

Conclusion

In summary, keeping a company's informational assets secure is always a significant concern. To secure the exchange of information between a client and server, the following issues need to be considered:

- That your message content is confidential and not available to others as it flows through different network environments, such as the Internet, intranet or an extranet. Encryption is used to ensure content confidentiality.
- The information exchange has integrity. There is an assurance that the messages received are indeed the same messages which have been sent. Encryption and digital signatures ensure integrity.
- There's accountability for both the sender and receiver. Digital signatures assure accountability (non-repudiation).
- Both parties in an exchange can be authenticated. Authentication ensures the identity of either party.

The SSL protocol addresses the following security issues.

- Confidentiality. After the symmetric key is established in the initial handshake, the messages are encrypted using this key.
- Integrity. Messages contain a message authentication code ensuring the message integrity.
- Authentication. During the handshake, the client authenticates the server using an asymmetric or public key.

Proxy servers are placed between a private and public network. They are used to control access to and from the private network by relaying information for authorized users and denying access to all others.

Other security components that require consideration are:

- Firewalls, which tend to be seen as protection between the Internet and a private network. Generally, speaking a firewall should be considered as a means to divide the world into two or more networks, one or more secure networks, and one or more non-secure networks.
- SOCKS servers, which are like proxy servers without the requirement for double connections. The user needs a SOCKSified client. The SOCKSified client directs its requests to the SOCKS port on the firewall. Sessions are broken at the firewall, as they are with proxy servers. With SOCKS, however, the connection to the destination application is created automatically once the user is validated. The SOCKS server acts as an application-level router between the client and the real application server.

Chapter 4
Performance

This chapter provides performance tips and topics that can improve your Domino Go Webserver site performance.

The Domino Go Webservers have a history of solid performance deployed as high-volume Web sites. For example, one such site at peak configuration consisted of 21 Domino Go Webservers. For that particular implementation, the Interactive Network Dispatcher, an IBM product, monitored and balanced the workload among these servers. The site handled up to 9 million hits a day with no reported performance problems. Or perhaps the goal is to improve Internet access performance. Then Domino Go Webserver could be implemented as a first-level caching server.

A site's effectiveness can be defined not only by the value and presentation of its content, but also by the number of requests received to obtain information. Users who are frequently disappointed with the length of time it takes to obtain information from a site will probably become less dependent upon that site, and search for alternative sources. From the user's perspective the only performance measurement that counts is the length of time required to receive a response from a Web site. The average user doesn't distinguish among nor understand the various Web site performance factors (for example, network, server, application design). Therefore, one significant attribute of a Web site's effectiveness is server response time.

But how is a Web server's performance measured? Let's start with a dictionary definition of performance.

> *The process of quickly and efficiently carrying through to completion.*

However, there are many factors that impact Web site performance; no single performance formula is applicable for all server implementations. Often these factors go beyond the ability to simply tune the server, and are related to either network or application performance. Nonetheless, those responsible for the server need to continuously evaluate the site's performance and make the appropriate changes when possible to improve its performance.

How Domino Go Webserver Works

Understanding the type of work Domino Go Webserver does helps when evaluating its performance. Let's start with a high-level approach to how the server works.

A Web server responds to requests for information received from clients by providing file objects via HyperText Transfer Protocol (HTTP) attached to a TCP/IP network. At a high level, the connection is made, the request is processed, a file system is accessed (or a CGI program is executed), the data is sent to the browser, and the connection is ended.

The HTTP server is busy serving files to the client upon request. The client, in many instances, is responsible for the processing and presentation of the content. There's a technique called *server side includes* which requires the server to process the content before it is sent to the client. Server side includes, when implemented, increase the server workload.

Domino Go Webserver listens on a TCP/IP well-known port (the default port is 80) for incoming requests. When an HTTP request arrives, Go reads it off the socket, decodes the request, extracts the URL from the requests, and maps it to a local file.

Domino Go Webserver can distinguish a CGI executable from a plain HTML document and take appropriate action. If the file is a CGI executable, a separate process executes the CGI script, and the results are returned to the Domino Go Webserver. Then the server writes the results to the socket to be sent back to the requester (usually a browser). If the request is for an HTML document, Domino Go Webserver searches its local memory cache. If the document is found, it is written to the socket. Otherwise, Domino Go Webserver reads the document from disk (an expensive operation), and then writes it to the socket.

This basic description implies that a well-tuned server must be able to efficiently read from and write to sockets, decode and manipulate strings, and search for and load files.

General Server Performance Issues

Examples of some of the different criteria that can be used to evaluate server performance are:

- Number of connections
- File size
- Peak connection rate

Note Often the term “hit” is used to refer to a single connection to a Web server.

To understand the performance issues affecting the Domino Go Webserver, review the different log files and activity events. The figure below shows the Domino Go Webserver activity events.

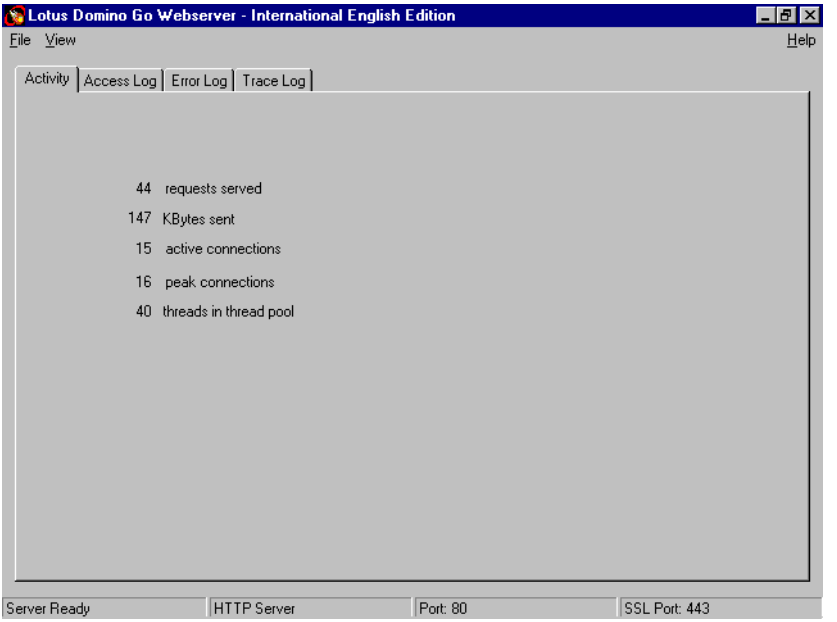


Figure 4.1. Activity Events Window

One of the activity events displayed by the Domino Go Webserver is the number of active connections. It is important to note that the number of active connections does not necessarily equal the number of active users. Domino Go Webserver supports HTTP 1.1 and persistent connections, which allows a single connection for transferring multiple data objects. Browsers can create multiple parallel connections to a server with the intent to improve end-user performance by simultaneously transferring multiple data objects embedded in an HTML file. For example, the Netscape browser permits the user to change the number of connections. However, a user that creates too many browser connections will probably negatively impact their performance.

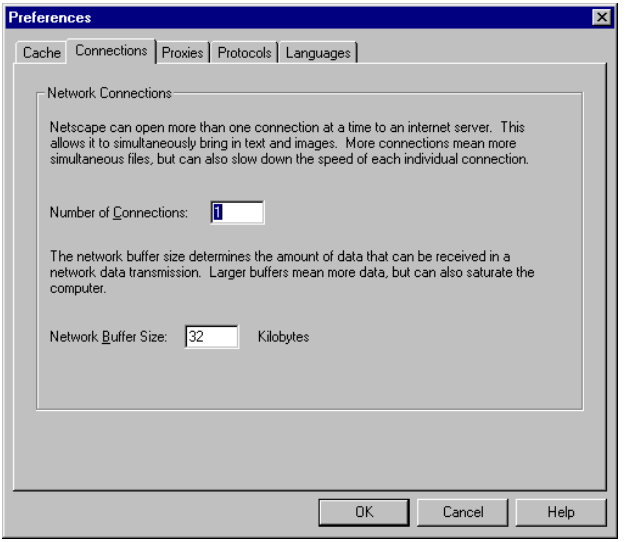


Figure 4.2. Netscape Connections Panel

More importantly, you should understand the relationship between the cause and event by working the server in a lab environment. The results of this type of work helps in recommending changes to improve server performance.

There are different test tools available for generating traffic to place a workload upon the server. These tools offer a method to benchmark server performance. Be familiar with the implementation of your choice of test tool so as not to be misled by any benchmarks based upon this type of work. For example, a certain benchmarking tool may only cover GET transactions, not POST transactions, that are used with forms and CGI programs. Nonetheless, these tools are useful and should be part of the overall strategy of determining the best performance-tuning mix for your particular Web server implementation.

The tuning tips contained within this chapter are intended as recommendations to be considered as part of a total performance plan to improve Web site performance. All tuning tips do not have the same impact in all environments. Use your judgment when deciding which tips to use for your Web server. When making any changes to improve performance, you should always make a backup copy of your server configuration file before you modify it. It's rarely a good idea to make any change before testing it within a lab environment. Nevertheless, even the best intended change may sometimes need to be removed quickly, especially when the law of unintended consequences comes into play. Any change that has to be removed may still be appropriate after further review and planning. So be flexible, do your homework and explain the performance value in the change to the others whose buy-in is necessary to move forward.

Domino Go Webserver Tuning Tips for Different Operating System Platforms

After installing Domino Go Webserver, you should tune the "httpd.cnf" configuration file to achieve peak performance. A properly tuned Web server significantly improves performance over the default configuration.

When performance-tuning the Domino Go Webserver, you can either directly edit the configuration file or use the product configuration and administration forms. See Appendix C for the default contents of the "httpd.cnf" configuration file.

The Performance form allows an administrator to change the following values, all of which affect server performance:

- Specify server priority to the operating system.
- Turn off the ACL file checking.
- Set a maximum number of threads.
- Set persistent connection time out.
- Set a maximum number of requests for a persistent connection.

The following two figures show the performance values that can be changed through the performance form.

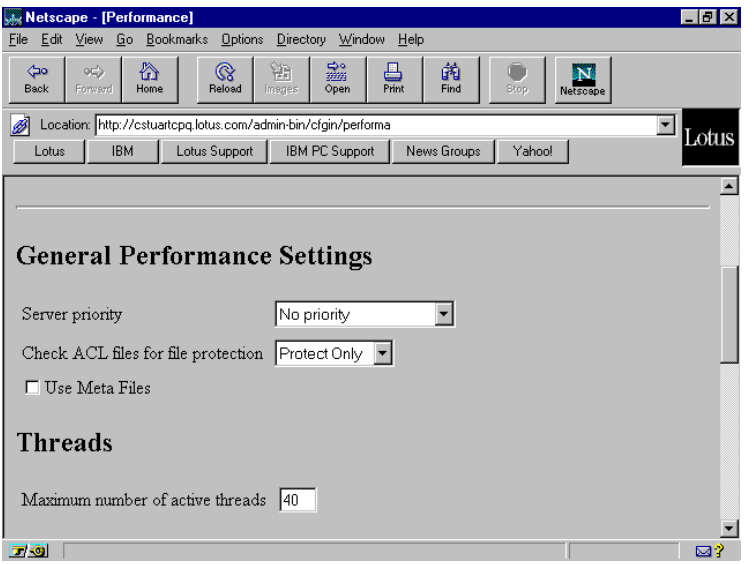


Figure 4.3. Performance Form

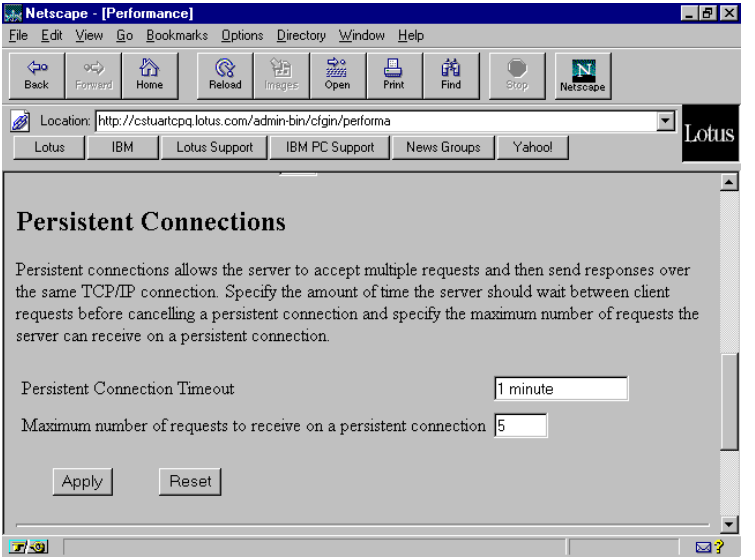


Figure 4.4. Performance Form Persistent Connections

See the *Domino Go Webserver Webmaster's Guide* for instructions on how to change server performance values.

Note The following tuning tips are not in any order of importance. Also, changes to the server configuration file require a server restart.

1. Place Frequently Referenced Files in Memory Cache

Static HyperText Markup Language (HTML) files remain constant for long periods of time. The contents can be served numerous times between updates. Static files that are frequently referenced may be cached in a server's memory.

When the server starts, it places specified files in a memory cache. Then, when a file is requested it can be fetched from the cache, rather than from a disk. When specifying the files you want cached, you might also need to increase the default maximum bytes and maximum file limits.

Tip To increase the memory cache size for maximum bytes and files, change the values associated with the following directives:

`CacheLocalMaxBytes`

`CacheLocalMaxFiles`

To load the most popular static files into the server's memory at startup, add the file names as values associated with the following directive:

`CacheLocalFile`

2. Suppress Time Stamp Check (for Memory Cached Files)

Each time a file in the server's memory cache is referenced, Domino Go Webserver checks the time stamp of the file on the disk to make sure the cached copy is current. If the site content does not change frequently, suppressing the time stamp check for files in the cache can improve performance.

Tip The time stamp check suppression value is set to off by default; changing the value associated with the following directive has performance implications:

```
LiveLocalCache   off
```

3. Turn Access Logging Off

By default the server records that a Web browser has accessed a file in a disk log. If you do not need this access logging, turn the logging off to improve server performance.

Tip Access logging is set to off by editing the configuration file and inserting a comment character "#" at the beginning of the following directive:

```
AccessLog
```

4. Suppress Access Logging from Selected IP Addresses or Host Names

If you do not need access logging for all requests, you can reduce the volume of data being logged and improve performance by using a log filter.

There are instances when you must log a subset of the total browser accesses. There are several methods to tell the server which accesses should not be logged.

Tip For example, to suppress logging from all browsers with IP address "9.*.*.*", list the suppressed addresses associated with the following directive:

```
AccessLog Exclude
```

5. Turn Agent Logging Off

Tip Agent logging is set to off by editing the configuration file and inserting a comment character "#" at the beginning of the following directive:

```
AgentLog
```

6. Turn Referrer Logging Off

Tip Referrer logging is set to off by editing the configuration file and inserting a comment character "#" at the beginning of the following directive:

```
ReferrerLog
```

7. Order Resource Mapping Tables

When a server begins processing a request, it searches the directives specified in the server configuration file. The server searches the "Pass" directives in the order specified within the configuration file. This search time can be reduced for specific URL sub directories by placing their corresponding "Pass" directives closer to the top of the "Pass" directives.

Tip To change the search order and improve search times for files, edit the directive statement:

```
Pass
```

Care must be taken with this approach. For example, if the very general "Pass" directive:

```
Pass / *
```

were moved to the top, then any subsequent more specific directives would be ignored.

8. Optimize Directory Listings

If you want your server to be able to return directory listings (by modifying the default "DirAccess off"), avoid some of the slower options.

Tip To return directory listings and maintain good performance, change the values with the following directives:

```
DirReadme              off
DirShowIcons           off
DirShowDate            off
DirShowDescription     off
```

9. Minimize Directory Depth

The server uses the file system when accessing a file. If the "Pass" directive being used for a particular file has a deep operating system directory structure, the file system opens and checks security (AIX read / write / execute, for example) at each level, each time the file is accessed. To reduce this processing, place the files in directories with a shallow directory structure. This will not help files that are already covered by "LiveLocalCache off," because the file system is not consulted.

Tip To reduce the file system subdirectory searching, place files in directories that are closer to the root directory.

10.Convert CGI to GWAPI

CGIs that execute impact server performance because the processing is completed on the server side, and the results are sent back to the browser. This capability is important to extend the business functionality of your server.

However, each CGI script requires significant server resources. A solution is to develop applications by using the Domino Go Webserver API to tie the application directly into the server. The GWAPI programs have a performance benefit over regular CGIs in that they are designed to specifically run a thread in the same address space as the Domino Go Webserver.

Tip You can convert your main programs to GWAPI programs for significant performance gain.

11. If Writing CGIs, Use a Compiled Language

CGI programs written in a compiled language like C or C++ are more efficient than interpreted script languages.

Tip Code CGI programs in a compiled language for performance gains.

12. Minimize Secure Operations

Encrypted sessions are important for extending your Web server's electronic commerce capabilities. The site's security requirements in many instances are a higher priority than performance considerations. Nonetheless, be aware that secure server operations take more CPU and elapsed time; use them only if required.

13. Do Not Install Security

Not all Web servers need security restrictions. If your site does not require security features, you may get a performance boost.

Tip If you are not using security, do not install it.

14. Shorten the Persistent Time Out

The original HyperText Transfer Protocol (HTTP) connection opened and closed a TCP/IP connection for every request, which consumed a significant amount of server resources to process the additional protocol overhead. HTTP fixed this problem by implementing persistent connections. Now the browser can keep the connection to the server open and make multiple requests, reducing both the network and server workloads. However, browsers that have completed transferring requested information can continue to maintain the connection. This connection is now idle and consuming server resources, such as operating system threads. To manage server resources and improve performance, tune the persistent connection time-out value.

Tip Edit persistent time-out value from "1 minute" to "10 seconds" by changing the value associated with the following directive:

`PersistTimeout`

15. Create Web Pages with Fewer Files

Server performance varies based on the type of file system. Each file system has different overhead and performance characteristics.

Tip Create Web pages with the fewest number of files.

16. Suppress DNS Lookup

If the domain name server (DNS) lookup is enabled, the server attempts to resolve IP addresses into fully qualified domain names for each access. This process greatly increases latency on requests, because the reverse DNS lookup (which must be completed before the request is completed) typically requires a new network connection to the name server. The setting of this server parameter affects the information entered into the access log. For example, if DNS lookup is set to off, then only the IP address is available from the access log.

Tip The DNS lookup value is set to off by default; changing the value associated with the following directive has performance implications.

```
DNS-Lookup        off
```

17. Tell Domino Go Webserver If Not Using Metafiles

Metafiles, which are sent in HTTP headers, specify extra information about a file. A typical metafile use might be to supply expiration dates for files. Metafiles are usually created by the user who creates the content and are managed with that content. Web servers that support metafiles must check for their existence each time they load a document.

Tip The metafiles value is set to off by default, changing the value associated with the following directive has performance implications.

```
UseMetafiles          off
```

18. Tell Domino Go Webserver If Not Using ACLs

Web servers use many different mechanisms to protect access to files and to store access control information. Domino Go Webserver uses two levels of access control. Entire directory structures can be protected with a protect directive and individual files within a protected directory structure can be protected with an access control list (ACL). ACLs typically provide more restricted access to certain files.

Tip The access control list value is set to never by default; changing the value associated with the following directive has performance implications.

```
UseACLs     never
```

Note The Domino Go Webserver 4.6 “readme” file specifies the following default value:

```
UseACLs     protectonly
```

19. Tell Domino Go Webserver If Not Using Server Side Includes (SSIs)

Server side includes (SSIs) are directives embedded in HTML documents and CGI programs that are processed by the server before returning the content to the browser. The performance cost related to SSIs is that the server must search every HTML document and CGI program output for the existence of embedded SSI directives, even documents that contain no SSI directives. If an SSI-embedded directive is found then it needs to be processed. This is a nice feature for processing dynamic content, but it creates significant server processing overhead.

Tip If not using SSIs, then tell Domino Go Webserver not to look for SSIs, change the value associated with the following directive:

```
Imbeds              off
```

Tip Tell Domino Go Webserver to process SSIs only on files with an extension of ".htmls" or "shtml."

```
Imbeds on SSIonly (not with ImbedsOnHTML)
```

Persistent Connections with HTTP 1.1

HTTP manages communication between browser clients and Web servers. The version commonly in use today is HTTP 1.0.

HTTP 1.0 is a stateless protocol which means that the Web server does not maintain any session information. Also, the connections between client and Web server are terminated after each request. The result is an increase in network traffic and server workload.

A single HTML document could consist of multiple data objects requiring multiple connections to retrieve the entire page (HTML document). Since HTTP 1.0 is a stateless protocol, the Web server needs to reconstruct the session for each subsequent request to retrieve the various data objects contained in the single HTML document. The additional overhead required to reconstruct the session impacts server performance.

The HTTP implementation of persistent connections significantly reduces the network overhead and server workload experienced with HTTP 1.0. The HTTP 1.1 design change permits a persistent connection between client and server for transferring multiple data objects contained within a single HTML document. The connection-oriented nature of the session eliminates the need for the server to reconstruct the TCP/IP session environment for every request. The benefit associated with HTTP 1.1 reduces server overhead as well as network traffic which leads to improved site performance.

The following figure illustrates how HTTP 1.0 processes an HTML document without using a persistent connection.

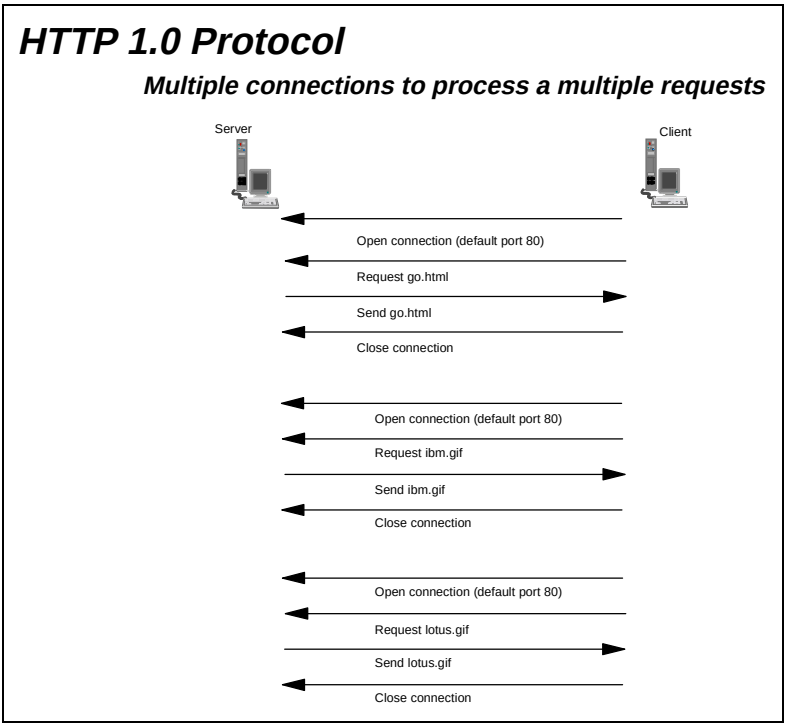


Figure 4.5.

When a client makes a request to a Web server for an HTML document, the following occurs:

1. A TCP/IP connection is established between client and server.
2. The client sends a request to the server for the document.
3. The HTML text is sent to the client.
4. The client interprets the HTML.
5. The client requests any imbedded data (for example, .gif files) from the server. Each one of these requests requires a separate connection. These requests can run both simultaneously and independently.
6. The client and server end the connection.

The server rather than the client, will normally close the connection. The server will wait a short period of time even if connection is idle before closing.

Persistent connections enable the client and server to establish a single connection that can be kept alive and used for multiple requests.

The client sends a request to the server which suggests “CONNECTION: Keep Alive.” On the first response, the server accepts or rejects the persistent

connection. If accepted, the client is able to send additional requests on the connection. Each request indicates whether to maintain or terminate the connection. The server can respond that it's maintaining or terminating the connection. "Well behaved clients" close connections that are no longer required.

The figure below is an illustration of HTTP 1.1 processing a document using a persistent connection.

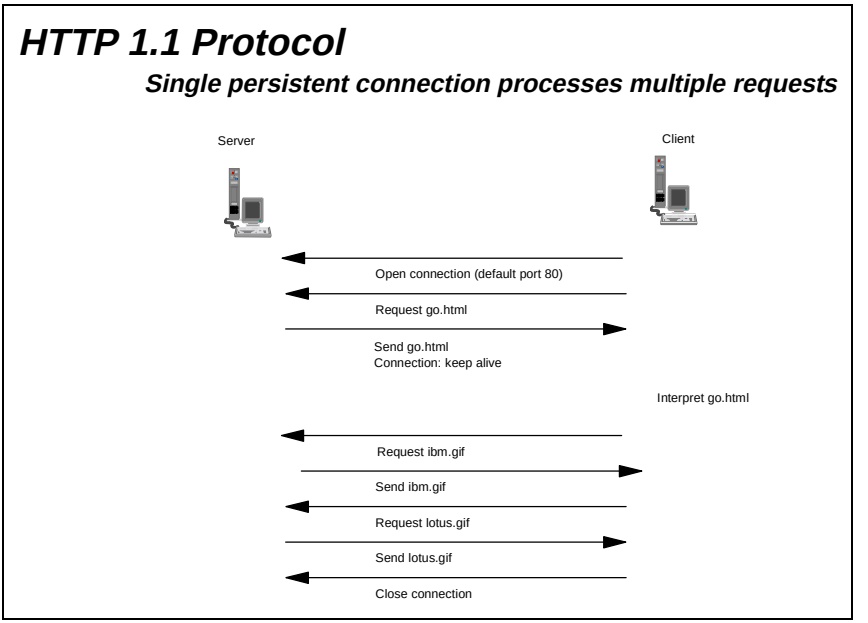


Figure 4.6.

To use persistent connections both the client and the server need to be HTTP 1.1-compliant.

Persistent connections will improve performance for smaller requests by requiring only one connection.

Pipelining

Because only one TCP/IP connection is established between the client and the server, there needs to be a mechanism to avoid sequential processing of the requests. Otherwise HTML data containing multiple requests may take an unacceptable amount of time to resolve.

Using HTTP 1.0, most browsers start multiple TCP/IP connections to speed up the retrieval of individual requests. However, this causes additional network traffic due to the connection setup overheads.

Pipelining permits a client to send multiple requests to a server without having to wait for each request to complete. Without pipelining, the client has to wait for each request to complete before sending the next request.

Pipelining may begin any time after the server agrees to a persistent connection in the first response header. Pipeline requests and responses may be packed into the same network buffers.

The figure below illustrates how HTTP 1.1 processes a HTML document using pipelining.

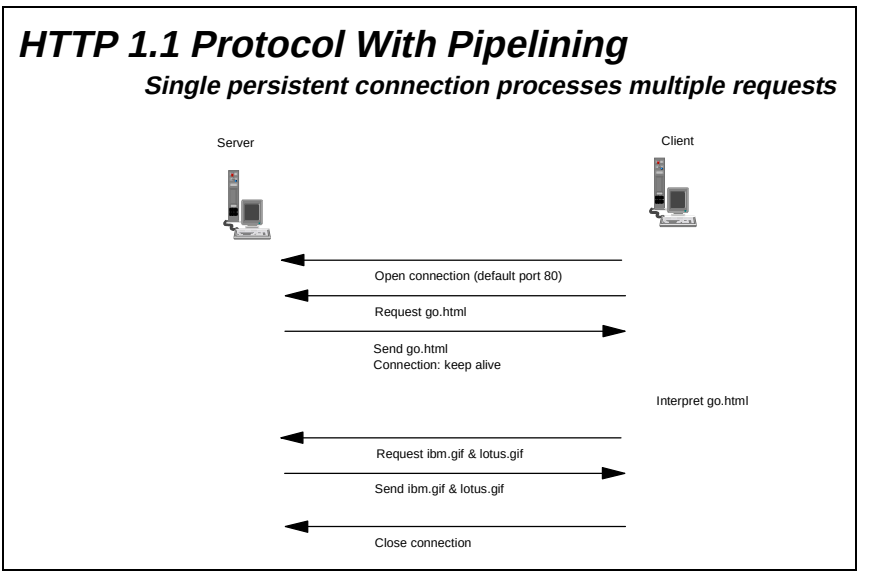


Figure 4.7.

The maximum number of requests that the Domino Go Webserver can process for each persistent connection at any one time is controlled by the following directive in the "httpd.cnf" configuration file:

"MaxPersistRequest"

The default value for "MaxPersistRequest" is set to five. Examine your HTML document to set this value accordingly.

The following directive coded in the "httpd.cnf" file specifies the length of time to wait before closing the TCP/IP connection.

"PersistTimeout"

Many clients will try to keep a connection alive even after a composite page is complete, and the user is performing some other type of activity. You should protect your server thread resources by limiting both the number of requests per connection, and the maximum time a thread will wait for another request on an idle connection.

The default "PersistTimeout" value is set to 1 minute; setting this to 10 seconds may be required for TCP/IP resource-constrained systems.

Tuning TCP/IP

TCP/IP is one of the first areas to address in order to improve Web server performance. A Web server is a TCP/IP socket application. When the server performs socket system calls, it consumes system resources.

A general practice is to maintain the current service levels of the TCP/IP protocol stack. Any TCP/IP protocol stack performance problems that have been corrected by the vendor are usually available in the current service levels.

Web servers and clients communicate using the TCP/IP protocol. There are some general rules for how a client and server connect and exchange information. A client and server transmit requests and responses. These request and response packets contain either data or commands. The commands manage the connection between client and server. Examples of some TCP/IP protocol commands are:

- **ACK** *Acknowledgement*
- **SYN** *Synchronize sequence numbers*
- **FIN** *No more data from sender*
- **RST** *Reset*

Then, there are the actual data contents of the file. TCP/IP exchanges these packets of information between the client and server. The TCP/IP packet size affects performance. TCP/IP packet size is influenced by many factors as it travels through the network. The TCP/IP packet size needs to be optimized for your particular environment. Minimizing the number of TCP/IP packets transmitted between client and server has a positive effect upon the server. For a detailed explanation of the TCP/IP protocol refer to RFC793.

A review of TCP/IP performance issues is beyond the scope of this document, however TCP/IP performance contributes directly to the overall Web site performance.

Network Dispatcher

Traffic intensity for Web servers is growing rapidly. The demand for system resources will go beyond the processing power delivered by high-end server platforms.

Performance solutions are evolving from single servers to multi-servers. Instead of placing all data on one server, it is replicated on multiple servers, creating a Web farm. The Web farm is controlled and monitored by software called a Network Dispatcher.

The Network Dispatcher routes TCP/IP-based requests to different nodes within a pool of servers, transparently to both client and server. The Network Dispatcher also uses load balancing to provide scalability.

When a browser sends a request to a single logical server, the request is intercepted by the Network Dispatcher and routed to the appropriate physical server.

The Network Dispatcher scheme offers better performance than symmetric multiprocessor (SMP) solutions. There may be multiple processors in a SMP machine, but essentially you have a single server whose scalability is hampered by a single TCP/IP stack, single operating system, and other similar issues. Collectively, these factors limit scalability.

Network Dispatcher improves scalability with negligible impact upon performance, by increasing the number of connections per second.

Conclusion

The goal is to continually improve server performance. The Domino Go Webserver processes every request, regardless of type, through the core server software. The current version of the Domino Go Webserver has shorter execution path lengths than earlier versions of the product. A shorter execution path means fewer machine instructions to process for certain types of function. Less processing improves CPU utilization, provides faster response time on individual browser requests, and increases overall server throughput.

Although the common tuning tips listed previously in this chapter apply to all operating system platforms, results may vary. Other tuning tips are unique to specific operating system platforms:

- AIX
- OS/2
- Windows 95
- Windows NT
- OS/390

Please see "Tuning Your Web Server for Better Performance," available on the Domino Go Webserver site (**www.lotus.com/dominogowebserver**), for detailed information.

Additional performance/capacity planning considerations are:

- Maximum connections per second the server can support.

  Identifying the daily peak-connection periods and the connection distribution pattern for the server over 24 hours. Please see "How Fast Can a Web Site Go?" (**`www.ics.raleigh.ibm.com/capacity`**), for detailed information.

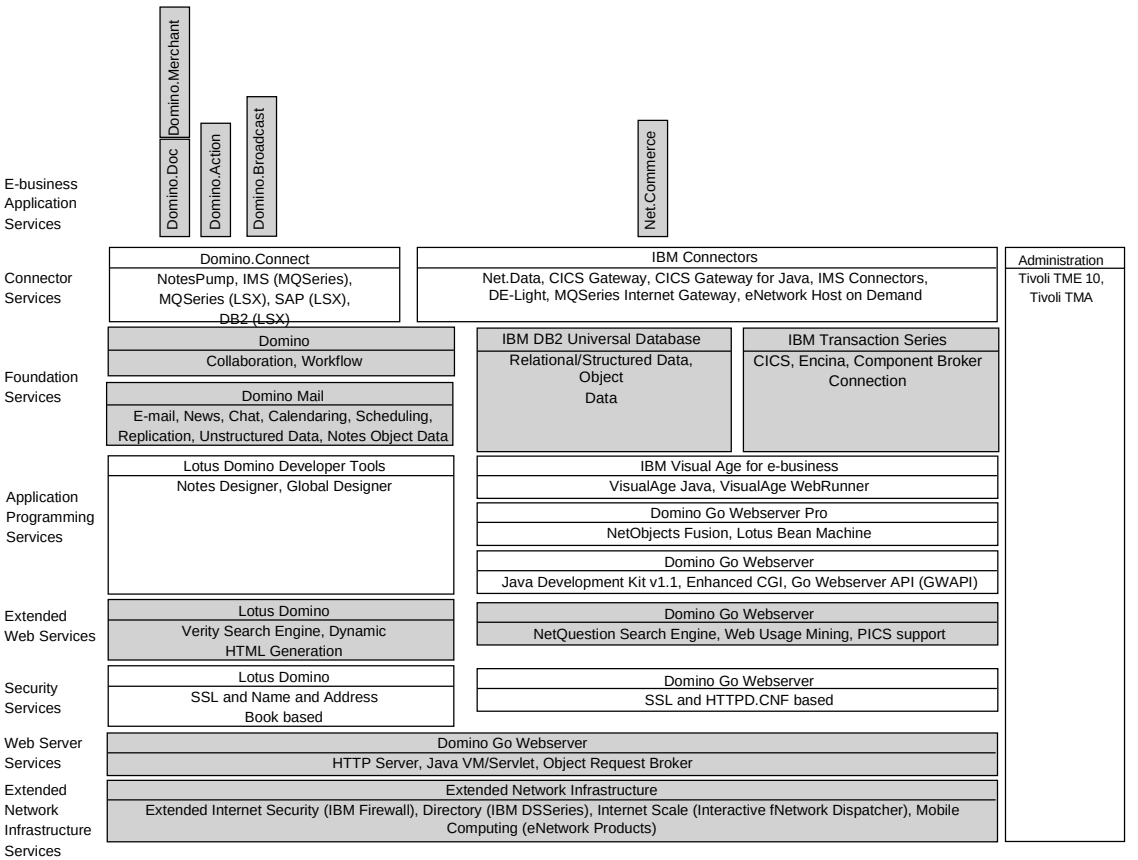
- The effect different file sizes have upon the server.

  The maximum connection rate depends upon the file sizes hosted by the server. Clients need to send acknowledgments to a server for receiving requested information. Larger files will receive more acknowledgments. Additional acknowledgments increase network traffic overhead, and subsequently slow server response. Please see "How Fast Can a Web Site Go?" (**`www.ics.raleigh.ibm.com/capacity`**), for detailed information.

Chapter 5
Application Development

This chapter will discuss the development of Web applications using the application programming services of Domino Go Webserver. We will also cover applications programming services provided by related products. In addition, we discuss some common Web programming models for developing Web applications, as well as the services that Domino Go Webserver and other products provide to support these models.

The following diagram illustrates the services and products highlighted in this chapter:



Web Programming Models

There are six dominant programming models currently in use on the Web:

1. Static HTML programming model.
2. HTTP CGI programming model.
3. Macro-based programming model.
4. Web client/server programming model.
5. Terminal emulation or HTML gateway model.
6. Dynamic generation of Web pages model.

Each of these models requires very different services and tools for application development. We will first briefly outline each of the models and list their characteristics.

A single Web application need not follow only one model; it may employ many programming models to provide a variety of functionality required in the application. Subsequent sections of this chapter will describe the tools that are referenced here in more detail.

1. Static HTML Programming Model

The static HTML programming model is the simplest of the programming models. Programming static HTML pages involves the editing of simple text files in HTML. Very few tools are required to support this programming model. While a simple text editor (like Notepad in Microsoft® Windows™), is sufficient, Domino Go Webserver Pro also includes the NetObjects Fusion product for generation and management of static HTML pages.

2. HTTP CGI Programming Model

The HTTP Common Gateway Interface (CGI) programming model provides a simple interface for running programs external to the HTTP server in a platform-independent manner. This interface has been in use by the World Wide Web since 1993.

A good document that describes this interface can be found at URL:

`http://hoohoo.ncsa.uiuc.edu/cgi/interface.html`

The purpose of CGI is to extend the capability of an HTTP server by providing a framework in which the HTTP server can interface with a program that is specified in a URL. The format of the URL allows parameters to be passed to the CGI program.

On the server side, the interface describes how the program is started by the HTTP server and how parameters for the program are passed using a combination of standard-input and environment variables. It also describes

how output information (usually HTML elements) are passed back to the HTTP server using standard output. Thus, in its simplest form, a CGI program can be defined as a program that:

1. Can be called as an executable and run as a child process of the HTTP server (this does not preclude the use of interpretive languages such as REXX and PERL; in these cases, their respective interpreters run as child processes of the HTTP server).
2. Is able to read from the standard input.
3. Is able to access environment variables.
4. Is able to write to the standard output.
5. Is able to access command line arguments passed to main.

CGI programs can be as simple or as complex as the programmer wants to make them, but clearly, the intent of the interface is to allow simple programs or scripts to be developed quickly.

3. Macro-based Programming Model

The macro-based programming model involves programming macros to generate Web pages. The products Lotus Notes Designer for Domino and IBM Net.Data both provide this type of development environment, providing a convenient mechanism for creating dynamic Web pages without complex programming.

To develop a Web application using this model, the user creates a dynamic script or formula that is interpreted by the macro processor. The macro processor is implemented as a CGI program, a persistent CGI program or a server API program, depending on the platform. The macro contains statements as well as macro directives that represent substitution variables, SQL calls, or program calls.

Macros can be used to build complex forms and process information from those forms. In Net.Data, the forms are defined with HTML as part of the macro script; in the Domino development environment, the forms are defined through a graphical interface.

4. Web Client/Server Programming Model

The second most prominent Web programming model used today is the Web client/server programming model. This model is not that different from the traditional client/server model that allows program logic to be distributed between a client and a server communicating over a shared protocol. The Web client/server programming model is different only in that Web protocols are used (that is, something that runs over TCP/IP) and the programming languages and associated tools are platform-independent (for example, Java, ActiveX).

What is attractive about the Web client/server model (compared to the HTTP CGI model) is that the program logic can be distributed as appropriate between the client and server, and a persistent connection is maintained which allows for long-running transactions. However, since the HTTP protocol and the HTTP server are not being used for communications, it is the responsibility of the client and server pair to agree on which application protocols are used and how issues such as security, encryption, integrity, and translation are handled. This becomes especially difficult when dealing with transient Java applets.

For the purposes of this document, the Web client/server programming model is defined as Java applications, or applets communicating to server applications over Web protocols. The server applications can be either new or existing procedural applications, or new Java servlets.

5. HTML Gateway Model

Today, although use of fixed-function "green screen" terminals, such as the 3270 and the 5250, is declining, the number of existing (sometimes called "legacy") interactive applications that work with a terminal interface remains very high. Consequently, interfacing with "green screen" applications will remain important for some time to come. The IBM CICS Internet gateway and Workstation Gateway provide these services for development of simple legacy-application Web development.

6. Dynamic Generation of Web Pages Model

To complete the list of Web programming models, we must remember the Lotus Domino programming model although it doesn't apply to Domino Go Webserver.

Domino allows application developers to use the powerful application development facilities of Notes to create Web applications. Notes includes its own graphical forms designer, directory services, replication services, messaging services, security, and workflow. Domino has the ability to dynamically generate HTML, which when combined with the services just mentioned, provides a true Web application platform. Domino and its Application Programming services are discussed in more detail in Chapter 7: Product Evolution.

Java Programming

In the previous and in the following sections of this chapter we often mention Java. In this section we will therefore define some Java terminology.

Java Programs
There are 3 kinds of Java programs:

- Applications
- Applets
- Servlets

Java applications are conventional programs that can be executed from a command line. In effect the Java Virtual Machine (JVM) is built into the program invocation.

A Java applet cannot be executed directly from a command line. It must be embedded in an HTML file (HTML version 3.2 or higher) and interpreted by the JVM in a Java-enabled browser, for example, Netscape Navigator or Microsoft Internet Explorer. This enables the applet to be portable across multiple platforms and operating systems without recompilation.

Java servlets are the equivalent of Java applets, but for the server. Servlets run on a Java-enabled Web server, for example, Domino Go Webserver, inside a JVM embedded in the Web server. They dynamically extend the services of the server by providing services over the Web using a request-response model.

Domino Go Webserver supports all versions of Java programming through the Java Development Kit 1.1 (JDK), which is included with the product.

JavaBeans

Domino Go Webserver Pro includes the Lotus BeanMachine for applet development by using JavaBeans.

The JavaBeans API consists of an architecture and a platform-independent API for creating applets and using dynamic Java software components. A JavaBean is a reusable software component that can be visually manipulated in builder tools like the JDK, which is included with Domino Go Webserver.

The JavaBeans API provides a framework for creating reusable Java objects that can interact with other objects dynamically. Specifically, a JavaBean is a Java class that obeys some strict protocols. It is these strict protocols that provide the reusability of the JavaBean.

Java Applet Communicating to CGI

We have talked about Java as a part of the Web Client/Server Programming Model, but another dimension that can be added to the HTTP CGI model is to make use of Java applets that communicate with CGI programs from the browser.

Java virtual machines built into browsers have mechanisms that allow its applets to access HTML documents and CGI programs directly. With this model, a CGI program may send HTML to a browser that includes a Java applet. The Java applet may use HTTP to retrieve additional information from the server, such as data stored in a file or output from another CGI program. This access is done independently of the browser session but using the HTTP services built into the browser. Since the HTTP protocol is being used, all of the advantages of HTTP are maintained, including its

authentication and encryption features. No changes are required to standard CGI or persistent CGI programs to support communications with a Java applet.

With this approach, logic can be split between the client and server with the client Java applet doing more than just providing an attractive window presentation. It can also dynamically receive information from server files and programs at the user's request.

Domino Go Webserver Application Programming Services

This section describes the tools for application programming that are provided with Domino Go Webserver and Domino Go Webserver Pro. Domino Go Webserver Pro contains Domino Go Webserver plus the Lotus BeanMachine and NetObjects Fusion to enhance application programming services.

Lotus BeanMachine

Lotus BeanMachine is a visual content assembly tool used to create Java applets. The tool is targeted at non-programmers who wish to create applets with multimedia, special effects, smart forms, and live data components. To produce an applet, users simply choose from a rich palette of reusable Java beans and java components and set a few properties.

Lotus BeanMachine is included in Domino Go Webserver Pro (an evaluation copy is also included with Domino Go Webserver).

NetObjects Fusion

NetObjects Fusion is a Web site builder that supports visual assembly of Web page content including multimedia, Java applets (perhaps produced using Lotus BeanMachine), and more. Its integrated custom templates generate a consistent look and feel. Navigation elements are created and updated automatically based on the site's structure. Fusion supports assembly of Web applications from content that was developed by IBM/Lotus or third-party tools.

NetObjects Fusion is included in Domino Go Webserver Pro (an evaluation copy is also included with Domino Go Webserver).

Extended CGI Support

Domino Go Webserver provides extended CGI support that lets you write CGI in Java as well as C, Perl, and other languages.

Java Development Kit

The Java Development Kit (version 1.1) is a set of programming tools. The following functions are provided in the Base Tools component of the product:

- **javac**: the Java Language Compiler that you use to compile programs written in the Java Programming Language into bytecodes.
- **java**: the Java Interpreter that you use to run programs written in the Java Programming Language.
- **jre**: the Java Runtime Environment that you can use to run Java applications. The jre is similar to the java interpreter, but is intended primarily for end users who do not require all the development related options available with the java tool.
- **jdb**: the Java Language Debugger that helps you find and fix bugs in Java programs.
- **javah**: this creates C header files and C stub files for a Java class. These files provide the connective glue that allow your code written in the Java Programming Language to interact with code written in other languages, for example, C.
- **javap**: this disassembles compiled Java files and prints out a representation of the Java bytecodes.
- **javadoc**: this generates API documentation in HTML format from Java source code.
- **appletviewer**: this allows you to run applets without a Web browser.

Other tools included in SDK are:

RMI Tools (ReMote Interface Tools): these tools implement the java.rmi remote interface.

Internationalization Tools: these tools convert non-Unicode Latin-1 (source code or property) files to Unicode Latin-1.

jar tool: this is a Java application that combines multiple files into a single JAR archive file. jar was designed mainly to facilitate the packaging of Java applets or applications into a single archive and improves download time to the client.

Java Servlets

Another way to build powerful and portable Web applications is to use Java on the server side in Java servlets.

Servlets are protocol and platform-independent server-side software components. Servlets run on a Web server machine inside a Java-enabled server (a server able to start the Java Virtual Machine) in order to support

the use of Java servlets. They dynamically extend the capabilities of the server because they provide services over the Web using the request-response model.

Note Domino Go Webserver Release 4.6 supports the JavaSoft Java Servlet API 1.0.

Further information on the JavaSoft Java Servlet API is available from:

`http://jserv.javasoft.com/products/java-server/sdk/index.html`

Servlets were introduced to interactively view and modify data and to generate dynamic Web content. From a high-level perspective, the Web content process flow would be:

- The client sends a request to the server.
- The server sends the request information to the servlet.
- The servlet builds a response and passes it to the server. That response is dynamically built and the contents of the response usually depends on the client's request.
- The server sends the response back to the client.

Servlets look like ordinary Java programs which begin importing some particular Java packages that belong to the Java servlet API. Since servlets are object bytecodes that can be dynamically loaded off the Net, we could say that servlets are to the server what applets are to the client. But, since servlets run inside servers, they do not need a graphical user interface (GUI).

Servlets and CGI-BINs

From a high-level perspective servlets can perform the same functions as CGI-BINs.

CGI-BIN applications tend to be harder to develop since you need technical knowledge on how to work with parameter passing. Generally the applications are not "fully" portable in that a CGI-BIN application written for a specific platform requires major changes to be able to run in a different environment. Some languages are also only supported in certain environments, for example, Visual Basic can only be used in a Windows environment. Each CGI-BIN application runs in a specific process that is activated by a client's request and is destroyed after the client has been served. This requires high startup, memory and CPU costs, and implies that multiple clients cannot be served by the same process.

On the other hand, servlets offer all the advantages of Java programs; they are portable applications and they are easier to develop. Servlets also allow you to generate dynamic portions of HTML pages embedded in static HTML pages using the servlet tag.

The main advantage of servlets over CGI-BIN applications is that a servlet is activated by the first client that sends it a request. It then continues running in the background, waiting for further requests, and each request generates a new thread, not an entire process. Multiple clients may be served simultaneously inside the same process, and typically the servlet process is destroyed only when the Web server is shut down.

One disadvantage of Java over CGI-BINs is that it is an interpreted language and its performance may not be as good as if it were compiled. Just-In-Time (JIT) compilers should resolve this problem.

Domino Go Webserver APIs

The GWAPI is an interface to the Go Webserver that allows you to extend the server's base functions. You can write extensions to perform customized processing, such as:

- Publishing customized pages based on the client's code level.
- Enhancing the basic authentication or replacing it with a site-specific process.
- Adding error-handling routines to track problems or alert for serious conditions.
- Detecting and tracking information that comes from the requesting client, such as server referrals and user agent code.
- Customizing security. For example, checking the IP address of each request to see if the same IP address keeps making requests using different names and passwords (indicating attempted unauthorized access).

The GWAPI compares favorably to other similar APIs (such as Netscape's NSAPI and Microsoft's ISAPI) for the following reasons:

- Efficiency

  The GWAPI is specifically designed for the threaded (rather than forked) processing used by Domino Go Webserver.

- Flexibility

  The GWAPI contains rich and compatible functions that can be used to clone other APIs (such as NSAPI, ISAPI).

  It is also platform-independent and language neutral. It runs on all the Domino Go Webserver platforms, and applications can be written in any of the programming languages supported by these platforms.

- Ease of use

  The GWAPI:

  - Passes simple data types by reference not value (for example, long *, char *).
  - Has a fixed number of parameters for each function.

- Has no return values from functions but does have a return code parameter.
- Includes bindings for the C language and provides an OS/2 GoServe compatibility module.
- Does not impact allocated memory.

Further information on the GWAPI can be found at:

http://www.ics.raleigh.ibm.com/pub/icswp003.html#HDROVERAPI

Web Site Application Programming Matrix

The following matrix summarizes the different types of Web models and indicates where the various application programming tools that we have been discussing may be best used:

Model	*Intranet*	*Extranet*	*Internet*
Simple	NetObjects Fusion	NetObjects Fusion, GWAPI	NetObjects Fusion, GWAPI
Interactive	NetObjects Fusion, extended CGI	NetObjects Fusion, GWAPI, extended CGI	NetObjects Fusion, GWAPI, extended CGI
Distributed	NetObjects Fusion, GWAPI, extended CGI, JDK, Lotus BeanMachine	NetObjects Fusion, GWAPI, extended CGI, JDK, Lotus BeanMachine	NetObjects Fusion, GWAPI, extended CGI, JDK, Lotus BeanMachine
Enterprise Distributed	NetObjects Fusion, GWAPI, extended CGI, JDK, Lotus BeanMachine	NetObjects Fusion, GWAPI, extended CGI, JDK, Lotus BeanMachine	NetObjects Fusion, GWAPI, extended CGI, JDK, Lotus BeanMachine

Connector Services Add-in Products

With the explosion of intranets and the Internet, there is a growing demand for access to dynamic Web pages and for applications to use transactions and data stored in large databases hosted on the enterprise systems.

The basic technologies to do this have been described earlier in this chapter. This section will review some of the products that can work with Domino Go Webserver to make it easier to achieve this goal.

The following products are discussed in this section:

- IBM's CICS Gateways
- IBM's IMS Internet Solutions
- DCE Encina Lightweight Client™
- IBM's eNetwork Host-On Demand

CICS Gateways

Many products allow users to access legacy applications from a Web browser or a network client, without any change to the applications code. We will mention some of these in the following section.

CICS/ESA Internet Gateway

The CICS Internet Gateway is a CGI server program that translates the input requests coming from the Web browser into the appropriate CICS External Presentation Interface (EPI) call. The call is then sent to the CICS server where it will schedule the appropriate transaction to be performed. It is as if it were a real CICS terminal requesting the transaction. During the first user request, the CICS server will create a terminal definition that will be associated with the user request. This permits the server to build a persistent session between the CICS server and the Web browser. The gateway will maintain knowledge of this session and will refer to the same terminal for all requests from that user. When the CICS Internet Gateway builds a session, it stores the state information as variables held in memory. This happens when the user starts the first transaction. The state information includes a session identifier that is placed in a hidden field on every HTML page sent to the browser, and the terminal identifier obtained during the first request sent to the CICS server. It is used in all subsequent EPI requests. A browser session will terminate, and all resources will be deleted, when any of the following occurs:

- The user clicks the Quit button.
- The user sends the CICS sign off transaction.
- A request has set AutoExit to ON.
- Any time-out values expire.

The CICS Internet Gateway supports both conversational and pseudo-conversational CICS applications and gives you access to CICS 3270 applications. It can't be used with applications that issue EXEC CICS START and EXEC CICS RETURN IMMEDIATE statements. In addition, there is no support for 3270 colors and text attributes and for 3270 numeric field attributes.

The CICS Internet Gateway is available on AIX, Windows NT and the OS/2 platform.

Further information on this capability is available on the Internet at the CICS Web page:

`http://www.hursley.ibm.com/cics/internet/index.html`

CICS Gateway for Java

The IBM CICS Gateway for Java provides a new and powerful way to access the transaction capabilities of CICS servers from the Internet. It combines the portable, architecture-neutral, object-oriented strengths of the Java programming environment with the power, integrity, and flexibility of CICS to bring access from the Internet to CICS applications running on a wide variety of server platforms. The CICS Gateway for Java enables Java applications to communicate with CICS servers through a CICS Client, and is itself a Java application.

Java uses TCP/IP to communicate, but the CICS client can communicate to CICS servers across SNA, TCP/IP, or NetBIOS. A single gateway can communicate through a CICS client over multiple protocols to multiple servers as required by the server.

Further information on this capability is available on the Internet at the CICS Web page:

`http://www.hursley.ibm.com/cics/internet/index.html`

IMS Internet Solutions

The IMS Internet Solutions (also called “IMS Connectors”) are the recently updated gateway products that enable users to easily access enterprise applications and data over the Internet using the Web browser.

Note Some products are still in beta version at the time of this writing (December 1997).

The tools provided are:

- IMS Web
- IMS WWW Templates
- IMS TCP/IP OTMA Connection (IMS TOC)
- IMS Client for Java
- IMS Object Connector

IMS Web

IMS Web allows you to provide access to IMS applications from clients across the Internet. Using information available from IMS, the tool generates the code necessary for mapping and communication between the Web and IMS.

IMS Web uses the TCP/IP communication infrastructure and middle-tier Web servers to make IMS transactions available from the Web. IMS Web also provides a preparatory tool as well as run-time environment code, and uses the IMS TCP/IP OTMA Connection for IMS access from the server.

This tool generates the following as a result of a number of intermediate steps:

- A CGI program for communicating from the Web to IMS.
- An input HTML form and the output HTML that displays the results of a transaction on the browser.

The latest release of IMS Web (Version 1.2) includes Net.Data.

IMS WWW templates

The IMS WWW templates provide sample and GWAPI programs to communicate with IMS transactions. These programs (written and supplied by IBM) are written in C, and call APPC routines to communicate with IMS.

IMS Client for Java

IMS Client for Java is part of the IBM Network Computing Framework for e-Business and provides templates for preparing a Java program to access IMS applications and data, running on an S/390.

IMS Client for Java consists of:

- Sample code to allow Java access from a workstation or other Java Virtual Machine environment, to IMS applications and data.
- A user-exit routine to translate messages into the format required by the IMS OTMA interface.

DCE Encina Lightweight Client

In addition to IBM's connectors, the DCE Encina Lightweight Client (DE-Light) by Transarc blends traditional static Web servers with commercial-quality or legacy applications.

The DE-Light Web Client provides access to existing DCE and Encina applications to desktop PCs and workstations with Web browsers.

The DE-Light Web Client is developed using the Java programming language. The browser downloads a Java applet from a Web page and executes it.

eNetwork Host On-Demand

eNetwork Host On-Demand is a new family of products developed in response to the request to obtain a Java terminal emulation capability to access host-based applications from within a Web browser.

eNetwork Host On-Demand Version 2

Host On-Demand V2 provides a thin-client solution that can be installed by a Web administrator on the Web server for download and execution. No code is required on the client machine, and the current version of the code is always downloaded. Because Host On-Demand is Java-based, it can run on any system with Java Virtual Machine (JVM) Version 1.1 support, including IBM's new network computers.

Application Programming Services Add-in Products

This section is intended to draw your attention to the application development options available to you. Many vendor tools are available but it is beyond the scope of this book to examine all the tools available in detail.

We will focus on Net.Data, the programming and query language included in Domino Go Webserver, and IBM VisualAge e-Business suite as examples of two programming tools.

Net.Data

Net.Data doesn't require considerable Web programming knowledge to easily transform static HTML Web pages into dynamic Web applications using Net.Data macros. Macros created for Net.Data combine the simplicity of HTML with the functionality of CGI applications, making it easy to add live data to static Web pages. Live data can include information stored in Net.Data-compatible databases locally or on other systems, REXX programs, C, and C++ programs, Java applets, or any other source such as more "traditional" high-level languages.

Net.Data Overview

The Net.Data macro language is an interpreted language. When Net.Data is invoked to process a macro, Net.Data directly interprets each language statement in a sequential fashion, starting from the top of the file. Languages that are not interpreted must be compiled into a program object before they are run. Any changes made to a macro can be immediately seen by specifying the URL that processes the macro without recompilation.

End-users of Net.Data Web macros see only the forms for their input and reports showing the results. A user fills out the forms, and points and clicks to submit the form. Net.Data processes the request as determined by the macro. The complete HTML page is dynamically built by Net.Data, and the result is returned to the browser.

The macro language was developed to be a simple and flexible language, enabling the application developer to use the full capabilities of HTML to create forms and reports. The application developer creates HTML source, SQL commands, and calls to other language environments, and stores them in macro files at the Web server. Variables are used to link the macro language commands and the HTML source within the macro file. The macro files are processed by Net.Data, and the results are passed to the HTTP server.

Features and Functions

Net.Data was designed with these objectives:

- To not require extensive programming by the application developer other than the use of HTML to create forms, SQL for queries and updates against the database, REXX for string manipulation and more advanced functions, and the Advanced Macro Language for ease-of-use functionality. For more information about REXX, see:

  `http://rexx.hursley.ibm.com/`

- To be flexible for a variety of Web applications that can grow from simple to complex.
- To be portable to multiple server platforms.
- To be usable with existing Web HTML editors and database query tools.

Another important advantage is that Net.Data supports many database formats, letting you work with data from a variety of data sources including DB2, Oracle, and Sybase databases on most platforms.

Advanced Macro Language

Net.Data Web macros combine familiar languages such as HTML, SQL, and REXX with a simple macro language. Capabilities of the macro language include:

- Multiple, named HTML sections in one macro.
- Ability to include files.
- If-Then-Else.
- Function definitions.
- Large library of predefined functions.
- Access to environment variables.
- Ability to disable the default report.
- Table variables.
- Ability to specify multiple paths for macros, executable, and includes.
- Persistent database connections.
- An external interface for platform-supplied language environments that can provide new sources to data.

Note For more information about the Net.Data macro language, see the online Net.Data Reference Guide at:

`http://www.software.ibm.com/data/net.data/docs/`

A Net.Data sample application is provided in Appendix C.

VisualAge for e-Business

VisualAge for e-Business is a new package of products for Web site development and includes:

- VisualAge for Java Enterprise Edition 1.0
- NetObjects Fusion 2.02
- Lotus BeanMachine 1.0
- VisualAge Webrunner
- DB2 Developers Edition 5.0
- Domino Go Webserver
- Netscape Navigator 4.04

NetObjects Fusion and Lotus BeanMachine are described in the Domino Go Webserver Pro Application Programming Services section of this chapter. We will discuss the other VisualAge for e-Business products in the sections that follow. While Domino Go Webserver Pro and VisualAge for e-Business both bundle NetObjects Fusion and Lotus BeanMachine, VA for e-Business is intended for professional programmers who might find themselves coding in Java. Domino Go Webserver Pro is intended for the developer who prefers a visual click and drag environment.

VisualAge for Java

VisualAge for Java is a powerful suite of application development tools that allows you to build complete 100% Pure Java applications, applets and JavaBeans by using the VisualAge "Construction from Parts" model.

VisualAge for Java enables you to extend your existing server applications to the Internet, intranet, or extranet rather than writing new Web applications from scratch.

VisualAge for Java simplifies your Java development process in four major ways:

- Client/server programming in Java is made easier through automatic generation of JavaBean components and middleware code that connects the Java client to existing transaction, data, and application servers.
- The development environment enables you to build scalable Java solutions that run on Windows 95, Windows NT, OS/2, AIX, OS/400, and OS/390.
- The version control facility allows you to easily go back to a working version of your code. Now this has been enhanced with a fully integrated repository-based team environment that encourages simple and easy collaboration and management of development projects.

- The project-based-development environment enables programmers to create Java applications / applets or Lotus JavaBean components using "Construction from Parts."

VisualAge for Java components are:

- IDE (Integrated Development Environment). IDE enables VisualAge for Java to deliver rapid application development to your Java program during the development phase. Version control facilities let you manage multiple releases of your Java classes. IDE keeps track of your source changes and automatically compiles it, ensuring that the latest updates to your source are reflected in the compiled applet.
- VisualAge for Java provides a Visual Composition Editor. This allows you to assemble applets, applications, and beans from pre-selected parts on the visual builder palette. You can drag the Java AWT (Abstract Windowing Tool) kit controls from the palette and visually drop them on the canvas to generate user interface Java code. Then you can connect the user interface to the business logic Java Bean. Since the Visual Composition Editor uses JavaBean components, you can easily extend existing Java applications to include additional controls.
- Enterprise Access Builders for Transactions. The Enterprise Access Builders generate JavaBeans components that establish fast connections between the Java client and CICS Transaction Servers. This allows your developers to focus on application logic instead of low level communications code by:
  - Partitioning client access across multiple server tiers.
  - Automatic creation of CICS External Call Interface middleware (ECI).
  - Automatic creation of code that changes Java client data types to COBOL CICS / ESA data types all of the above, packaged into a JavaBean component for easy connection to the Web user interface using the Visual Composition Editor.
- Enterprise Access Builders and Data VisualAge for Java allows database manipulation through Java Language. The JDBC standard defines a common base on which other data access tools can be built. The Enterprise Access Builder tools generate JavaBean components that create, retrieve, update, and delete database information. This means that the application can:
  - Use methods / data members of generated objects to query databases. Results of query can be stored and manipulated as Object data structures (collections).
  - Executes SQL.
  - Gain direct access via JDBC to DB2 on AIX, OS / 2, NT.

- Gain direct access to JDBC data sources (Sybase, Oracle).
- Access through JDBC and DDCS to DB2/400 and DB2/ESA.

All of the above can be packaged into a JavaBean component.

- Enterprise Access Builders for Applications. VisualAge for Java allows for the selection of the best language for the job. You can build applications that connect Java clients to existing or new applications on the server.

VisualAge for Java lets you add a class, add a method, or change a method while still in the test phase. The modified code is compiled and inserted into the application, without the need to exit the debugger, find and fix the source code, regenerate the Java bytecode, and then navigate back to the original error to validate your fix.

VisualAge for Java has a single-user development repository. The single-user development repository provides the ability to store code, as well as import and export files from the repository to classic file systems.

VisualAge for Java will incorporate a repository-based source control mechanism that enables multiple developers to work concurrently on an application while reducing the number of costly source code collisions. The system also keeps track of the levels of source that make up an application as well as with version control implemented at the class level.

VisualAge WebRunner Toolkit

VisualAge WebRunner Toolkit, a product from Taligent, Inc., is a natural add-in product for VisualAge for Java and JDK 1.1 for developers of applications built from 100% pure industry standard JavaBeans and Servlets.

The VisualAge WebRunner Toolkit includes these features:

- WebRunner Bean Tools
- WebRunner Servlet Tools
- WebRunner Beans.

An upcoming new technology, called BeanExtender Technology, is currently available for free download as a technology preview from IBM's alphaWorks Web site.

WebRunner Bean Tools is a set of tools for development of enterprise-class JavaBeans. WebRunner Bean Tools currently includes the following products:

- **WebRunner Bean Wizard** builds a Bean quickly and easily creates 100% pure JavaBeans with powerful built-in functionality. Once you've created your Bean, WebRunner Bean Tester helps you evaluate and test it. WebRunner Bean Tester provides an environment for testing of both Bean, and Bean interaction.

- **WebRunner Migration Assistant** converts your ActiveX or OCX controls to JavaBeans. The Migration Assistant analyzes the interfaces of your existing controls and generates skeletal Beans. All you do is add the internal logic. This tool reduces ActiveX conversion time dramatically.
- **WebRunner Bean Works** is a sophisticated framework or class library designed to give developers the power to create maintainable JavaBeans.

WebRunner Servlet Tools enable you to create fast, portable server-side applications and Servlets. These tools include:

- *WebRunner Server Works for Java* is a set of Java frameworks that provide a simple, consistent API for servlet access to host and Web server functions across platforms and server API sets. It conforms to and expands JavaSoft's Servlet API.

WebRunner Beans is a set of Java Beans for quick assembly of applications. WebRunner Beans includes these Beans:

  - **WebRunner Network Beans** are for access to Internet file transfer, mail, and news protocols (FTP, POP3, SMTP, NNTP).
  - **WebRunner User Interface Beans** is a set of Java Beans that provides application developers with user interface widgets.
  - **WebRunner Gauge Beans** is a set of Java Beans that allow users to view data in a variety of ways through gauges, indicators, and other data-visualization beans.

Application Programming Add-in Product Matrix

In summary, the following table represents the application programming add-in products as they relate to the Web site models discussed in Chapter 1: Architecture.

Model	*Intranet*	*Extranet*	*Internet*
Simple			
Interactive	Net.Data	Net.Data	Net.Data
Distributed	Net.Data, VisualAge for Java, VisualAge Webrunner	Net.Data, VisualAge for Java, VisualAge Webrunner	Net.Data, VisualAge for Java, VisualAge Webrunner
Enterprise Distributed	Net.Data, VisualAge for Java, VisualAge Webrunner	Net.Data, VisualAge for Java, VisualAge Webrunner	Net.Data, VisualAge for Java, VisualAge Webrunner

With the exclusion of the static Web sites, application languages like Net.Data may be used in every environment, while more complete and complex tools tend to suit distributed environments more.

e-Business Application Add-in Products

When a new technology enters the market, there is a time lag before new solutions that take advantage of that technology are developed. This was true in the past for both transactions systems and the client/server environment and it is true now for network solutions.

Today, network applications tend to be extensions of existing transactions or client/server oriented solutions but the forecasts show that the development of completely new e-Business applications as one of the fastest areas of growth in Information Technology.

Net.Commerce

Net.Commerce is the IBM solution for electronic commerce, designed particularly for a network environment. To build this type of application, you can either build your own solution based on a Web server product such as Domino Go Webserver, or you can use Net.Commerce. Net.Commerce tends to be used when catalog and transaction volumes grow and more sophisticated merchandising tools are required.

Net.Commerce Technical Capabilities

The following section provides an overview of the major components of the Net.Commerce System:

- **Web Server Functions**. DGW establishes secure sessions across the Internet between the end user's browser and the Net.Commerce applications using the SSL2 protocol. Customers can also use the Netscape Enterprise Server if they prefer.
- **Scalability and High-Availability Features**. Net.Commerce is scalable from a laptop all the way up to clusters of IBM RS6000 AIX machines and S/390 mainframes (and has been announced as a Product Preview for the AS/400 platform).
- **On-the-fly Web Page Generation**. Net.Commerce Server, Daemon and Server Director functions can generate Web pages on-the-fly during a customer session, for example, by pulling product information and prices from the database and converting the result into Web pages that customers see on their browsers.
- **Performance features**. Net.Commerce stores the dynamic pages in a cache after they have been created. When the data reflected in a page changes, the dynamic pages are regenerated. The result is improved performance.

- **Database Functions** are IBM DB2 or Open Database Connectivity (ODBC). Net.Commerce comes with IBM's DB2 because DB2 runs on all the platforms supported. Net.Commerce supports the ODBC standard which gives access to ODBC-enabled databases in addition to DB2.
- **Data Import Function**. A Data Import Utility allows the seller to load catalog and other data in bulk into the Net.Commerce database. DB2 facilities help to keep the data in Net.Commerce synchronized with a source product database, if needed.
- **Macro Language Function**. Net.Commerce uses Net.Data's macro language and software drivers to manage connections between databases and information servers, and to change standard processes.
- **Customization Functions**. Net.Commerce comes with an extensive set of customization tools (Commands and APIs).
- **Staging Function**. Staging support enables Web administrator to test updates before going live to uncover errors or problems before users see them. The tested database updates are explicitly promoted into operational status.

IBM Net.Commerce Special Features

Special features enhance the value of Net.Commerce to large or complex organizations or those with very large catalogs:

- **SET-Enablement**. Net.Commerce is SET-enabled and will take advantage of CommercePOINT Payment, IBM's implementation of SET secure credit card payment, when VISA and MasterCard release the Internet version of the protocol.
- **Friendly and Secured Buyer interface**. For example, the credit card number remains hidden from seller employees.
- **Configuration Interface**. With CommercePOINT eTill, a seller can configure different acquirer gateways for each type of credit card they accept.
- **Administration Interface**. The administration tool in CommercePOINT eTill will provide sellers with basic reporting capabilities, transaction reversals, and other administrative functions.
- **Catalog Assistant**. The Catalog Assistant tool enables the seller to create catalog interaction metaphors that buyers use to select goods.
- **Product Assistance**. The Product Assistance tool helps the seller create an online catalog experience which is targeted to the buyer that understands the product group (such as PCs) and wants to select a specific product based on features (for example, RAM, chip, hard disk space). Product research starts with a click on the "most important" product feature that the buyer requires, and products that do not match this criteria are eliminated.

- **Sales Assistance**. The Sales Assistance tool helps the seller create catalogs that allow customers to easily find the product or service that best suits their individual and unique needs although they are not familiar with the products' features, taking the buyer through a series of questions about how the product will be used.
- **Product Comparison**. The Product Comparison tool enables the seller to choose which product attributes will appear on a side-by-side comparison page. The page can display two, three or four products with differences noted in a contrasting color or font. Buyers will see the comparison after the number of candidates has been whittled down through the Product Assistance and Sales Assistance processes.

e-Business Application Add-in Product Matrix

The following table summarizes the connector services add-in products that we have discussed in this section as they relate to the Web site models discussed in Chapter 1: Architecture.

Model	*Intranet*	*Extranet*	*Internet*
Simple			
Interactive		Net.Commerce	Net.Commerce
Distributed		Net.Commerce	Net.Commerce
Enterprise Distributed		Net.Commerce	Net.Commerce

Although some functions of Net.Commerce may also be used in an intranet environment, Net.commerce and the other business applications are generally designed for external networks.

Extended Network Infrastructure Add-in Products

A real-world environment may be very complex, with many Web servers distributed in the network and many other different servers for services like mail, transaction processing, database serving, and so on.

New communications technologies already available today, are forecasted to grow rapidly in the next years.

In this section we will describe some solutions available for managing different servers, performance auditing, systems availability, and security assurance in complex environments.

IBM Firewall

A network firewall is a protective barrier that controls the flow of information between a trusted (internal) network and an untrusted (external) network. A firewall lets users in your internal network use resources from an external network such as the Internet without compromising your network data and other resources.

Today, institutions and companies want to take advantage of the many services and resources available on the Internet. And so, they want to connect their private, secure networks to the open, non-secure Internet.

How do you give your users access to the vast store of information on the Internet while protecting your network and data from unauthorized access? Use a firewall to provide a blockade between a secure, internal, private network, and another (non secure) network like the Internet. It protects your internal network from other networks on the Internet, while at the same time letting TCP/IP applications (such as e-mail and WWW) access hosts beyond your network, on the Internet.

Three keys ways that a firewall can protect your network are filtering, proxy servers, and encryption. The IBM Firewall provides all three services. Domino Go Webserver implements only the proxy functions as described in Chapter 3: Security.

IBM Firewall Filtering

Filters are one way the IBM Firewall controls traffic from one network to another. The filters operate on criteria such as IP source or destination address range, TCP ports, UDP responses, Internet Control Message Protocol (ICMP) responses, and TCP responses.

IBM Firewall as an Application-Level Proxy

The IBM Firewall application-level proxy is referred to as the proxy server. If a proxy server does not prompt a user for a password or other authentication, it is considered transparent. The IBM Firewall implements full proxy servers for Telnet and FTP as well as transparent proxy servers for Telnet, FTP, and HTTP.

A full proxy server is a secure server that runs on the firewall and performs a specific TCP/IP function on behalf of a network user. The user contacts the proxy server using one of the TCP/IP applications (Telnet or FTP). The proxy server makes contact with that remote host on behalf of the user, thus controlling access while hiding your network structure from external users. The IBM Firewall FTP and Telnet proxy servers can authenticate users with a variety of authentication methods, including password verification, SecurID cards, S/Key, and SecureNet Key cards.

IBM Firewall as a Circuit-Level Proxy

The IBM Firewall implements circuit-level proxies in two ways; as a SOCKS server and through network address translation (NAT).

The SOCKS server can intercept all outbound TCP/IP requests that would cross between your network and the Internet. The SOCKS server provides a remote application program interface so that the functions executed by client programs in secure domains are piped through secure servers at the firewall workstations, hiding the client's IP address. Access is controlled by filters that are associated with the SOCKS rules.

The SOCKS server is similar to the proxy server. But while the proxy server actually performs the TCP/IP function at the firewall, the SOCKS server just identifies the user and redirects the function through the firewall. The actual TCP/IP function is performed at the client workstation, not at the firewall. (This saves processing in the firewall.) The users in the secure network can use the many TCP/IP products that support the SOCKS standard.

The other implementation of circuit-level proxy is network address translation (NAT) which can be used for both TCP- and UDP-based applications.

With the explosive growth of the Internet, IP address depletion becomes a problem. NAT provides a solution. The IBM Firewall manages a pool of IP addresses that can be used to communicate on the Internet. NAT translates secure IP addresses to temporary, external registered IP address from the address pool. This allows trusted networks with privately assigned IP addresses to have access to the Internet. This also means that you don't have to get a registered IP address for every machine in your network.

Both the SOCKS server and NAT effectively hide your internal IP addresses from the outside world.

IBM Firewall Use of Encryption

The IBM Firewall provides secure communication across a public network like the Internet through virtual private networks (VPNs). A VPN is a group of one or more secure IP tunnels. When two secure networks (each protected by a firewall) establish a VPN between them, the firewalls at each end encrypt and authenticate the traffic that passes between them. Likewise, when a VPN is established between a remote client and a firewall, the traffic between them is encrypted and authenticated. The exchange of data is controlled, secure, and validated.

Note For more information, see:

`http://www.ics.raleigh.ibm.com/firewall`

Internet Scale

Internet Scale is available from IBM as part of its Network Computing Framework. To Webmasters and administrators, Internet Scale is the distributed, load-balanced and global page delivery system for e-Business applications. Internet Scale gives companies the ability to expand secure Web access to the enterprise so extranet as well as intranet users can retrieve file pages with high performance and reliability.

As Web site volumes grow, organizations struggle with the need to expand capacity, manage what they have, and improve service. Adding Web servers, as can be done with Domino Go Webserver, provides a first step to increasing capacity, but usually moves the bottlenecks at the Web servers to other places.

Internet Scale provides the necessary components to streamline and optimize Web browser to Web server communication as well as Web server to file page data.

Internet Scale is a collection of products that can be used:

- Individually to improve specific elements of a Web site.
- Collectively to provide a comprehensive solution for delivering information via the Web.

Internet Scale includes:

- **IBM's Interactive Network Dispatcher**

  Interactive Network Dispatcher provides load-balancing across a group of Web servers by providing routing mechanisms to treat multiple computers as a single Web site.

- **Transarc's DFS Distributed File System**

  DFS provides a coherent, scalable, manageable, and secure file service so Web servers can be distributed regardless of the location of the file pages they use. In addition, DFS automatically caches the most frequently delivered pages at the Web server to increase network performance.

The Interactive Network Dispatcher receives and routes the requests to the most available local or remote Web server. The Web server then uses the DFS global file sharing system to retrieve the requested Web file pages and send them back to the users via the Web environment.

Note The InterScale solution doesn't apply to all the platforms supported by Domino Go Webserver.

Interactive Network Dispatcher

Is a TCP Connection router and load-management software that supports multiple back-end servers. It also allows busy Web sites to increase capacity by linking many individual servers connected to a single logical server. The Interactive Network Dispatcher, operating in a 2-tier balancing scheme, provides scaling and load balancing at a local cluster of servers and across many geographic locations to address the Web server needs of large enterprises and high-volume e-Business sites. The Interactive Network Dispatcher integrates with a domain name server, as part users of the load-balancing solution in a WAN environment, and functions independently for load balancing within a local area network.

Within a WAN environment, the Interactive Network Dispatcher transparently balances network load among a pool of TCP/IP servers. As a group of servers able to serve the same Web pages, pools can be configured flexibly by the customer and may overlap in some way.

Within the LAN environment, the Interactive Network Dispatcher provides an integrated cluster-port-server load-balancing mechanism for TCP applications.

For more information, see:

http://www.ics.raleigh.ibm.com/netdispatch

Transarc's DFS

DFS provides caching, file replication, and system machine-independent unique file names. DFS caches information on Web servers, reducing both server and network load, which sustains performance under high loads. With DFS, files can be automatically replicated across multiple servers that make up a single Web site. This ensures consistency of data while achieving the scalability and availability required by large, distributed Web sites.

In addition, every DFS file across the system has a unique name, independent of the physical location of the file.

When used with its DFS Web Secure feature, DFS provides an additional level of security by enforcing user authentication at the Web browser level.

For more information, see:

http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/DFS/index.html

Extended Network Infrastructure Add-in Matrix

The following table represents the extended network infrastructure services add-in products as they relate to the Web site models discussed in the Chapter 1: Architecture:

Model	*Intranet*	*Extranet*	*Internet*
Simple		IBM Firewall	IBM Firewall
Interactive	eNetwork Products	IBM Firewall, eNetwork Products	IBM Firewall, eNetwork Products
Distributed	eNetwork Products, Internet Scale	IBM Firewall, eNetwork Products, Internet Scale,	IBM Firewall, eNetwork Products, Internet Scale,
Enterprise Distributed	eNetwork Products, Internet Scale, DFS series	IBM Firewall, eNetwork Products, Internet Scale, DFS series	IBM Firewall, eNetwork Products, Internet Scale, DFS series

Summary

Development Tools Matrix

The following table represents a summary of the tools and add-in products discussed throughout this chapter and is intended to help you to focus on the solutions, tools, and products you can use to develop a Web site.

Model	*Intranet*	*Extranet*	*Internet*
Simple	• Static HTML • NetObjects Fusion	• Static HTML • GWAPI • NetObjects Fusion • IBM Firewall	• Static HTML • GWAPI • NetObjects Fusion • IBM Firewall

continued

Model	*Intranet*	*Extranet*	*Internet*
Interactive	• Static HTML • HTML Gateway • CGI-Bin • Macro-based • Dynamic Generation • NetObjects Fusion • extended CGI • CICS Internet Gateway • IMS Web • eNetwork Host On-Demand Release 1 • Net.Data • eNetwork Products	• Static HTML • GWAPI, HTML Gateway • CGI-Bin • Macro-based • Dynamic Generation • NetObjects Fusion • extended CGI • CICS Internet Gateway • IMS Web • eNetwork Host On-Demand Release 1 • Net.Data • Net.Commerce • IBM Firewall • eNetwork Products	• Static HTML • GWAPI • HTML Gateway • CGI-Bin • Macro-based • Dynamic Generation • NetObjects Fusion • extended CGI • CICS Internet Gateway • IMS Web • eNetwork Host On-Demand Release 1 • Net.Data • Net.Commerce • IBM Firewall • eNetwork Products
Distributed	• Static HTML • GWAPI • CGI-Bin, Macro-based • Web Client/Server • Dynamic Generation • NetObjects Fusion • extended CGI • JDK • Lotus BeanMachine • CICS Gateway for Java • IMS Internet Solution • DE-Light • eNetwork Host On-Demand Version 2 • Net.Data • VisualAge for Java • VisualAge Webrunner • eNetwork Products • Internet Scale	• Static HTML • GWAPI • CGI-Bin • Macro-based • Web Client/Server • Dynamic Generation • NetObjects Fusion • extended CGI • JDK • Lotus BeanMachine • CICS Gateway for Java • IMS Internet Solution • DE-Light • eNetwork Host On-Demand Version 2 • Net.Data • VisualAge for Java • VisualAge Webrunner • Net.Commerce • IBM Firewall • eNetwork Products • Internet Scale	• Static HTML • GWAPI • CGI-Bin • Macro-based • Web Client/Server • Dynamic Generation • NetObjects Fusion • extended CGI • JDK • Lotus BeanMachine • CICS Gateway for Java • IMS Internet Solution • DE-Light • eNetwork Host On-Demand Version 2 • Net.Data • VisualAge for Java • VisualAge Webrunner • Net.Commerce • IBM Firewall • eNetwork Products • Internet Scale

continued

Model	*Intranet*	*Extranet*	*Internet*
Enterprise Distributed	• Static HTML, GWAPI, CGI-Bin, Macro-based, Web Client / Server , Dynamic Generation, NetObjects Fusion, extended CGI, JDK, Lotus BeanMachine, CICS Gateway for Java, IMS Internet Solution, DE-Light, eNetwork Host On-Demand Version 2, Net.Data, VisualAge for Java, VisualAge Webrunner, eNetwork Products, Internet Scale, DFS series	• Static HTML, GWAPI, CGI-Bin, Macro-based, Web Client / Server , Dynamic Generation, NetObjects Fusion, extended CGI, JDK, Lotus BeanMachine, CICS Gateway for Java, IMS Internet Solution, DE-Light, eNetwork Host On-Demand Version 2, Net.Data, VisualAge for Java, VisualAge Webrunner, Net.Commerce, IBM Firewall, eNetwork Products, Internet Scale, DFS series	• Static HTML, GWAPI, CGI-Bin, Macro-based, Web Client / Server, Dynamic Generation, NetObjects Fusion, extended CGI, JDK, Lotus BeanMachine, CICS Gateway for Java, IMS Internet Solution, DE-Light, eNetwork Host On-Demand Version 2, Net.Data, VisualAge for Java, VisualAge Webrunner, Net.Commerce, IBM Firewall, eNetwork Products, Internet Scale, DFS series

Chapter 6
Administration

The aim of this chapter is to explain the administration of Domino Go Webserver and to provide some hints and tips that may help you in managing your system. For a detailed description of administration please refer to the *Webmaster's Guide*.

Configuration and Administration Forms Page

After installing Domino Go Webserver, the home page of the server gives you a link to the Configuration and Administration Forms main page. It is recommended that, since configuration and administration tasks are not supposed to be done by all clients, one of the first tasks that you perform is to make this link available only to the administrators. You can access the Configuration and Administration Forms page directly by entering the following URL:

```
http://<system_name>/admin-bin/initial
```

From the main page of the Configuration and Administration Forms, you can link to each of the input forms.

Basic Function

The Basic Function entry in the Configuration and Administration Forms page is used in managing the basic parameters of a Web server. Most of them are acquired during the installation process and, in the simplest environments, you may never need to change them.

The basic parameters include:

1. Domain name or the IP address of the Web server.
2. IP port of the Web server which will listen for client requests.
3. Directory where the server was installed.
4. The option of whether to retrieve the host name of the clients or simply to use their IP address.
5. The choice of whether the server binds to only one IP address (that specified in parameter 1) or to all IP addresses available on the machine.
6. The choice between different kinds of service side include that can be developed on a Web server.

We will not describe the use of each of these parameters, since this is covered in the *Webmaster's Guide* and in the help page of the interactive utility, but we will highlight some issues here that relate to security and its possible effect on performance.

Name or IP address?

The domain name or the IP address of the Web server are functionally equivalent. If you use the domain name, you will also need a Domain Name Server (DNS) to resolve the address.

Client Name Lookup

With option 4 set to:

- **Off**

  Clients are identified only by IP address and therefore:

  - You cannot use host names in resource mapping directives, you must use only an IP address.
  - The server will use only client IP address in log files.

- **On**

  Clients are identified by host name after a call to a DNS and therefore:

  - You can use host names in resource mapping directives but you cannot use an IP address.
  - The server will use client host name in log files.

  Note Depending on the workload on the server, carefully evaluate the performance of the Web server and of the DNS. Every call to the Web server must wait until the Web server asks the DNS for the client's host name and receives the answer from the DNS.

IP Port

You must use a port number that is different from the default (80) when you need to have more than one Web server on the same machine and on the same IP address. Every server must have a different port number.

Note If you want use a port number that is different from 80 for security reasons, avoid choosing other frequently used numbers like 8000, 8008 or 8080 or similar numbers.

The following example shows selected pieces of the configuration files of three Domino Go Webservers running on the same machine.

In this example, clients can enter the URL without indicating the port number. The GOWEB1 server, listening on port 80, will redirect requests for the other servers.

GOWEB1 Basic Directives

```
Hostname goweb1.lotus.com

DNS-Lookup     Off

Port           80
```

GOWEB1 Resources Mapping

```
Exec  /admin-bin/*     C:\DWG\Admin\*             goweb1.lotus.com
Exec  /cgi-bin/*       C:\DWG\CGI-Bin\*           goweb1.lotus.com

Pass  /reports/java/*  C:\DWG\HTML\reports\java\* goweb1.lotus.com

Pass  /reports/*       C:\DWG\HTML\reports\*      goweb1.lotus.com

Pass  /Docs/*          C:\DWG\Docs\*              goweb1.lotus.com

Pass  /httpd-internal-icons/*  C:\DWG\Icons\*     goweb1.lotus.com

Pass  /icons/*         C:\DWG\Icons\*             goweb1.lotus.com

Pass  /wsApplet/*      C:\DWG\Applets\*           goweb1.lotus.com

Pass  /Admin/*.gif     C:\DWG\Admin\*.gif         goweb1.lotus.com

Pass  /Admin/*.html    C:\DWG\Admin\*.html        goweb1.lotus.com

Pass  /common/*        C:\WEB\COMMON\*            goweb1.lotus.com

Pass   /*              C:\DWG\HTML\*              goweb1.lotus.com

Redirect /*   http://goweb2.lotus.com:8000/*      goweb2.lotus.com

Redirect /*   http://goweb3.lotus.com:8080/*      goweb3.lotus.com
```

Note The Pass directive has been added in order to manage all common functions as error management pages only once in the Web site.

GOWEB2 Basic Directives

```
Hostname goweb2.lotus.com

DNS-Lookup     Off

Port           8000
```

GOWEB3 Basic Directives

```
Hostname goweb3.lotus.com7

DNS-Lookup     Off

Port           8080
```

Note In these examples GOWEB2 and GOWEB3 servers have their own set of Pass directives, including a “common” Pass directive and no Redirect directives.

User Administration

There are two administration tools available with Domino Go Webserver for user administration. The first one is the *htadm* command described in the *Webmaster's Guide* and the second is *The Administration of Users* option in the interactive utility *Configuration and Administration Forms.*

The two tools are not completely equivalent and the use of the interactive utility is compulsory if you want to use some security features of Domino Go Webserver.

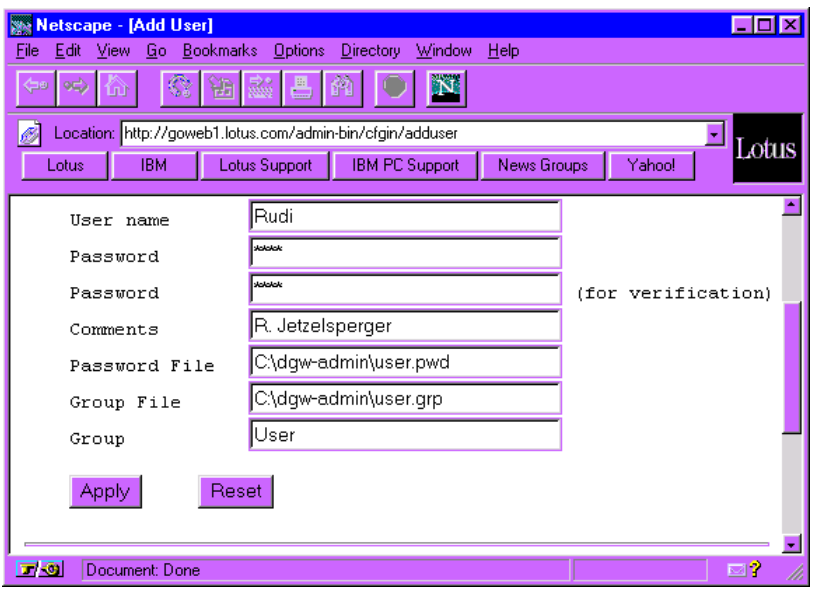
Both tools support the following functions:

- Adding a user
- Deleting a user
- Change a user's password
- Checking for a user's existence

The *htadm* command has one more function to create file where archive users and passwords, the interactive utility self-creates new files.

Adding a User

The add user action serves to explain the difference between the two tools. The following picture shows the screen of the interactive administration tool for adding a user.



The last two fields are exclusive features of the interactive utility that allow users to be entered into a group. This feature is not available with the *htadm* command.

The Group identification may be used in the security-related directives of the configuration file. This means that if you need to use Group Id for security reasons you must use the interactive utility for managing users. Alternatively you use the *htadm* command to add the users and then manually edit the group file with a text editor.

We recommend that you choose one of the two tools and then always use the same tool in order to minimize errors in files which can affect security.

In servers where there is also active FTP, or the server is a file server or performing a file-sharing function, it is important that password and group files:

- Are not allocated in shared, public or well-known directories (such as WINNT or WIN95 on Windows systems).
- Are backed up regularly.

Directories and Welcome Page

The earliest type of browsing on the Internet provided a graphical interface of server directories. This ability nowadays tends to be used in only a few Web sites, for example, in help desk applications, as an easy and inexpensive way to store and find frequently added, deleted or changed files.

The main disadvantage of directory browsing is that you can only partially map or hide the directory structure and the contents of your server. For information on how to configure the directory browsing function, see the manual or on line help. You will also find some recommendations relating to performance-tuning this function in Chapter 4: Performance.

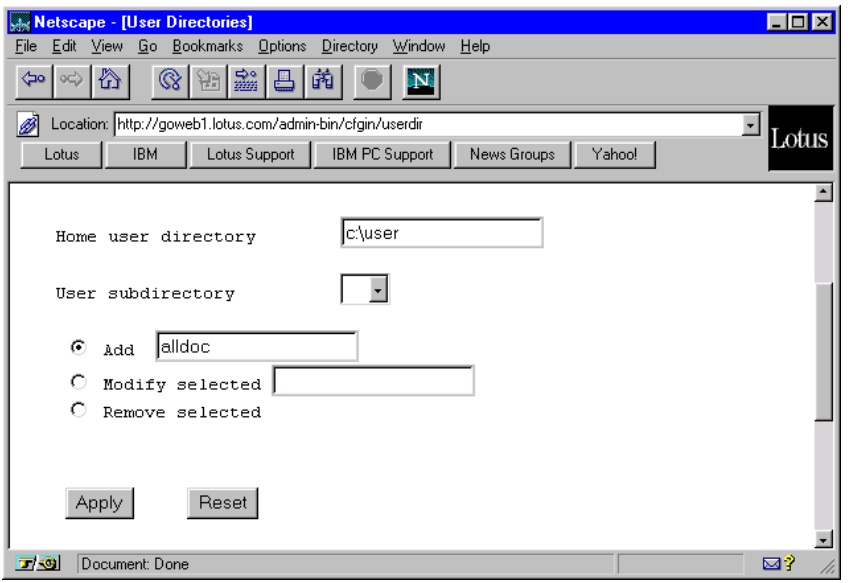
User Directories

One interesting use of the directory browsing function is to enable user directories to be customized. In order to browse user directories, you must enter a URL, for example:

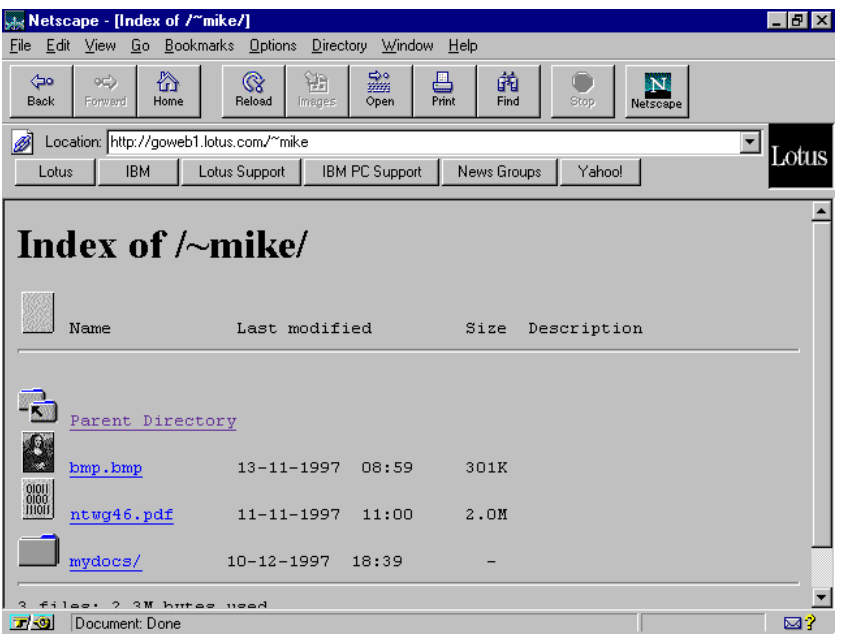
`http://goweb1.lotus.com/~username`

This URL points to a subdirectory named *username* of the home user directory defined in the User directories page.

In this example, suppose you want your users to be able to browse a subdirectory named *alldoc* in their own directory; the home user directory is *c:\user*:



This next figure shows what the user will see as result of a request for the user *mike*.



Welcome Pages

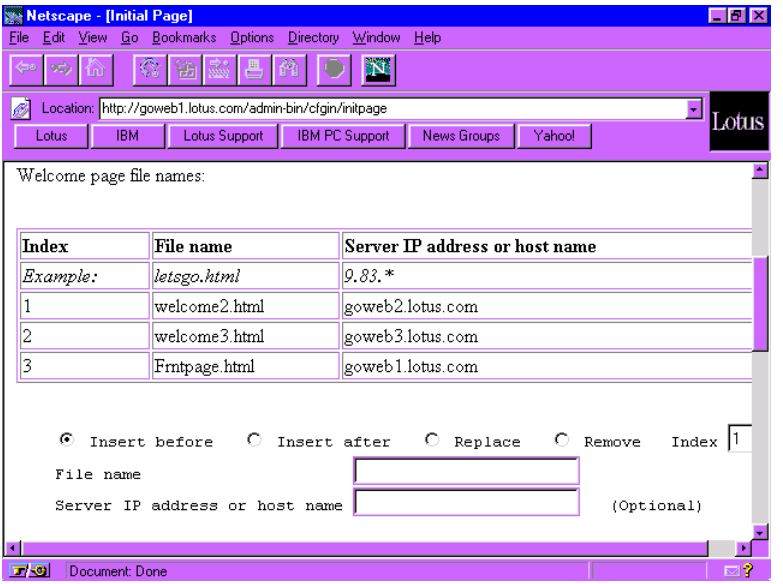
In the interactive administration tool, directory management directives are grouped with the directives for welcome or home HTML pages.

You may have as many welcome pages as you need. This ability to have multiple welcome pages is useful if you want to host multiple Web sites on the same Web server (sometimes known as a Web hotel).

Multihosting Example

The reasons for hosting multiple Web sites on the same Web server have already been described in the Installation chapter of this book. In the following example we will repeat the example from the previous section, but with three Web sites hosted within a single Web server instead of three individual Web servers.

This is what you will see when you enter the Directories and Welcome Page function and then select Initial Page:



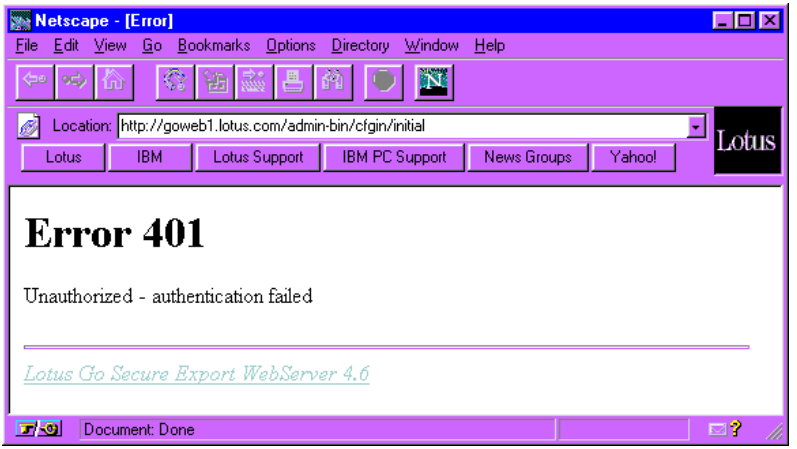
Every Web site will also have a complete set of resource mapping directives. The default resource mapping of Domino Go Webserver, customized for a multihost environment, would be something like:

```
Pass   /common/*        C:\WEB\common\*

# - #

Exec   /admin-bin/*     C:\DWG\Admin\*              goweb1.lotus.com

Exec   /cgi-bin/*       C:\DWG\CGI-Bin\*            goweb1.lotus.com

Pass   /reports/java/*  C:\DWG\HTML\reports\java\* goweb1.lotus.com
```

```
Pass  /reports/*       C:\DWG\HTML\reports\*      goweb1.lotus.com
Pass  /Docs/*          C:\DWG\Docs\*              goweb1.lotus.com
Pass  /httpd-internal-icons/*  C:\DWG\Icons\*     goweb1.lotus.com
Pass  /icons/*         C:\DWG\Icons\*             goweb1.lotus.com
Pass  /wsApplet/*      C:\DWG\Applets\*           goweb1.lotus.com
Pass  /Admin/*.gif     C:\DWG\Admin\*.gif         goweb1.lotus.com
Pass  /Admin/*.html    C:\DWG\Admin\*.html        goweb1.lotus.com
Pass   /*              C:\DWG\HTML\*              goweb1.lotus.com
# - #
Exec  /cgi-bin/*       C:\DWG2\CGI-Bin\*          goweb2.lotus.com
Pass  /reports/java/*  C:\DWG2\HTML\reports\java\* goweb2.lotus.com
Pass  /reports/*       C:\DWG2\HTML\reports\*      goweb2.lotus.com
Pass  /Docs/*          C:\DWG2\Docs\*             goweb2.lotus.com
Pass  /httpd-internal-icons/*  C:\DWG2\Icons\*    goweb2.lotus.com
Pass  /icons/*         C:\DWG2\Icons\*            goweb2.lotus.com
Pass  /wsApplet/*      C:\DWG2\Applets\*          goweb2.lotus.com
Pass  /Admin/*.gif     C:\DWG2\Admin\*.gif        goweb2.lotus.com
Pass  /Admin/*.html    C:\DWG2\Admin\*.html       goweb2.lotus.com
Pass   /*              C:\DWG2\HTML\*             goweb2.lotus.com
# - #
Exec  /cgi-bin/*       C:\DWG3\CGI-Bin\*          goweb3.lotus.com
Pass  /reports/java/*  C:\DWG3\HTML\reports\java\* goweb3.lotus.com
Pass  /reports/*       C:\DWG3\HTML\reports\*      goweb3.lotus.com
Pass  /Docs/*          C:\DWG3\Docs\*             goweb3.lotus.com
Pass  /httpd-internal-icons/*  C:\DWG3\Icons\*    goweb3.lotus.com
Pass  /icons/*         C:\DWG3\Icons\*            goweb3.lotus.com
Pass  /wsApplet/*      C:\DWG3\Applets\*          goweb3.lotus.com
Pass  /Admin/*.gif     C:\DWG3\Admin\*.gif        goweb3.lotus.com
Pass  /Admin/*.html    C:\DWG3\Admin\*.html       goweb3.lotus.com
Pass   /*              C:\DWG3\HTML\*             goweb3.lotus.com
# - #
Pass   /*              C:\WEB\common\*
```

Error Message Customization

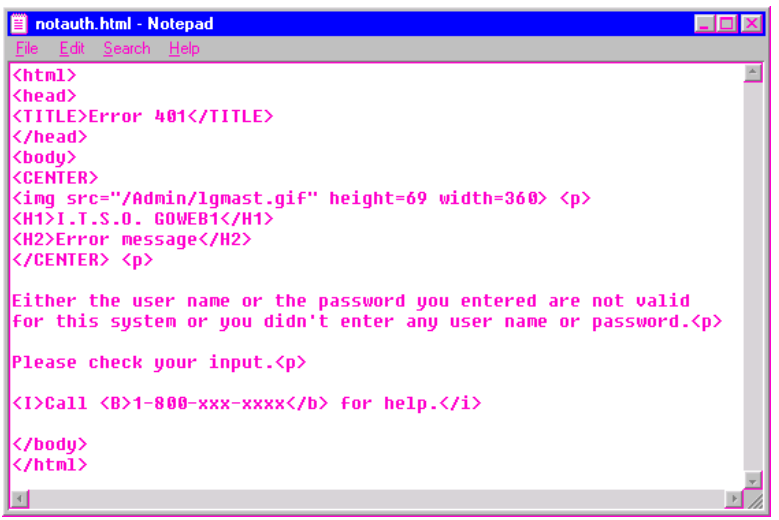
As a rule, the error management in HTTP is very limited and the message text unfriendly. For example, an error message will consist of an error number and a short and sometime cryptic description, as in the following example:



However, you can easily build a simple and more friendly error message in HTML and display this instead of the standard message. The following section describes the steps to achieve this.

Step 1. Build the New Message

With an HTML editor, create a new message for the HTTP error (in this case we will create one for error 401) that contains the new message text. This particular error (401) is caused by either the user name and password sent by the client not being valid for this request or by the client not sending a user name and password at all.

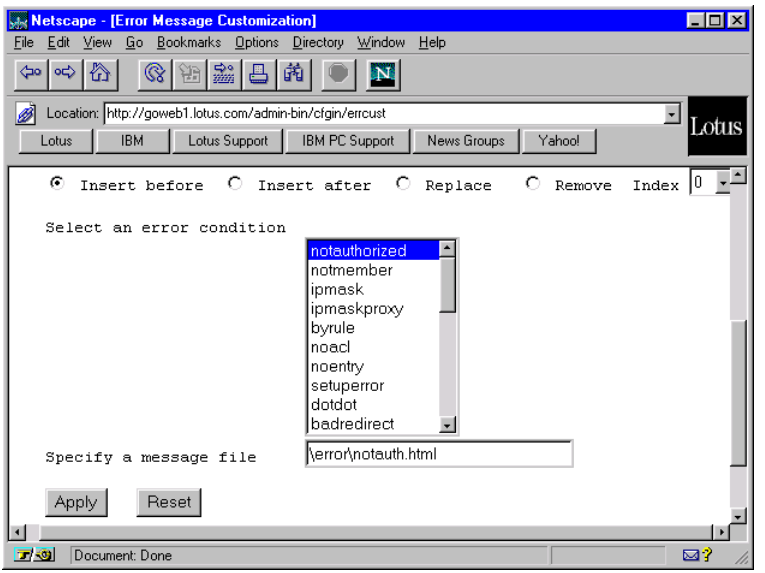


Save the HTML file in the subdirectory *error* in the directory *HTML*, which is the default directory of Domino Go Webserver.

Step 2. Create an Error Message Directive

You must now link the HTML page to the error message function of the Web server.

In the Configuration and Administration Forms section, choose the Error Message Customization entry and link the HTTP error to your new HTML page.



Step 3. Test

Connect from any browser to your Web server and try to access a protected page. Enter an unauthorized user name or password; the result should be:

This result is surely more friendly and user-oriented than the standard HTTP message and does not take much time to implement.

Logging and Reporting

The Domino Go Webserver can create several types of logs:

- Access logs
- Agent logs
- Error logs
- Cache access logs (if the server is a caching proxy)
- CGI error logs
- Referrer logs

For access logs, you can set filters and create reports that help you analyze the information in the access logs. For error logs, you can set certain maintenance options. For the other types of logs, you can specify the name of the log and where you want it to be filed. Also, the information contained within certain logs can be used to create customized reports. Each day at midnight, the server closes the logs for that day and creates new logs.

The default configuration file log names and file path values are:

AccessLog \WWW\Logs\httpd-log

AgentLog \WWW\Logs\agent-log

ReferrerLog \WWW\Logs\referrer-log

ErrorLog \WWW\Logs\httpd-error

CgiErrorLog \WWW\Logs\cgi-error

The following configuration file directives and their default values affect the format of the logs:

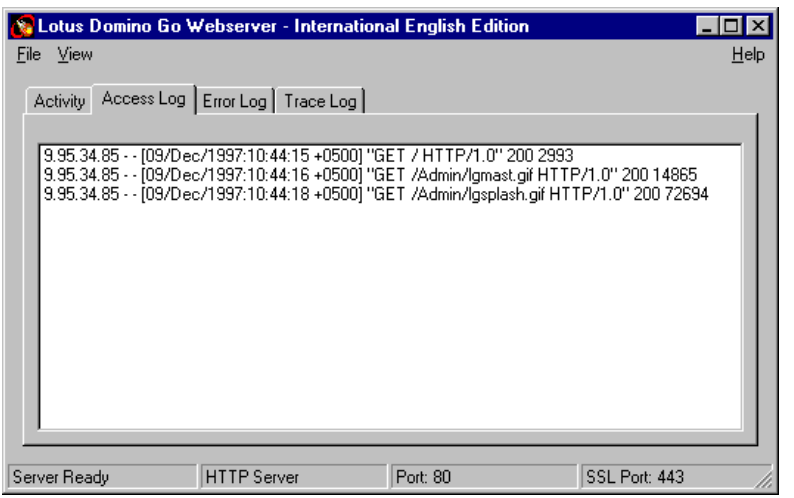
LogFormat Common

LogTime LocalTime

LogToGUI off

The "LogToGUI" directive is set "off" by default to optimize server performance. This directive displays log entries in the Domino Go Webserver log windows.

An example of Domino Go Webserver log with the "LogToGUI" directive "on" follows:



If the "LogToGUI" directive is set "off", then the specific log that requires analysis needs to be located within the "\WWW\Logs" subdirectory and reviewed. For example:

```
9.95.34.85 - - [08/Dec/1997:10:07:31 +0500] "GET / HTTP/1.0"
200 2993
```

```
9.95.34.85 - - [08/Dec/1997:10:07:34 +0500] "GET
/Admin/lgsplash.gif HTTP/1.0" 200 72694
```

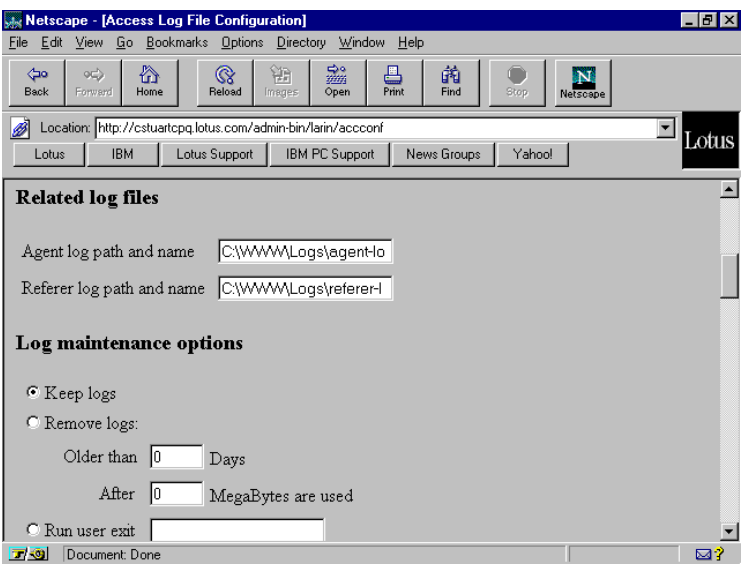
```
9.95.34.85 - - [08/Dec/1997:10:08:42 +0500] "GET / HTTP/1.0"
200 2993

9.95.34.85 - - [08/Dec/1997:10:08:43 +0500] "GET
/Admin/lgmast.gif HTTP/1.0" 200 14865
```

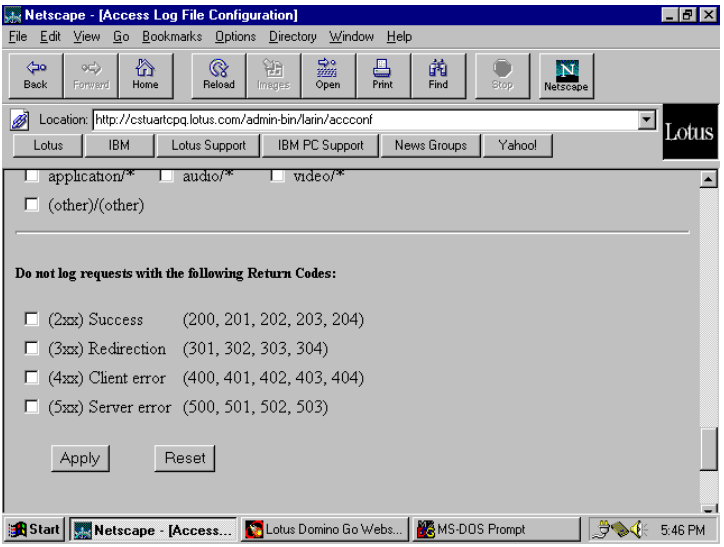
Access, Agent and Referrer Logs

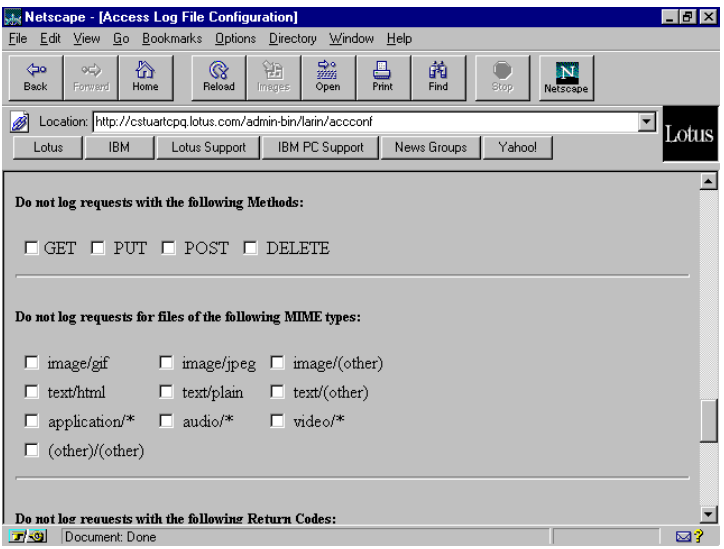
The access log stores a record of server activity. For example, for each access an entry is created within the access log showing:

- What was received
- When it was requested
- Who requested it
- The method of the request
- The type of file that your server sent in response to the request
- The return code, which indicates whether the request was honored

The following figures show the access log configuration form showing the different fields where log maintenance options can be specified.







Log maintenance allows you to specify how to handle the accumulation of daily logs. You can choose to keep old logs, remove logs after a certain age, or run a your own program to handle the logs.

Note The directives specified for access logs apply to both agent and referrer logs, as well. However, when maintaining access logs based on the collective size, the collective size of all access logs only is used to determine when actions will be performed and the size of the agent or referrer logs are not included in the calculation.

To remove access log files based on age, change the following configuration file directives:

AccessLogArchive **purge**

AccessLogExpire **number-of-days**

To remove access log files based on collective size, change the following configuration file directives:

AccessLogArchive **purge**

AccessSizeLimit **number-of-megabytes**

The agent log indicates which Web browser was used to access a Web page. The referrer log identifies which Web browser was used to access a Web page that referred (or linked to) the requested Web page. By default the server writes an entry to the agent and referrer logs each time a client sends the server a request. For every entry made in the access log:

- The agent log has a corresponding entry that indicates the browser used to display the page or file requested by the client.
- The referrer log has a corresponding entry that indicates the referring page.

Error Logs

The error log includes errors encountered by the server's clients, for example, timing out. The CGI error log contains standard error output (stderr) from CGI programs. Also, if the server is running as a proxy, the server can create two different type of logs:

- A proxy access log, which contains access requests for files that come from the proxy server.
- A cache access log, which contains access requests for files that come from the proxy server's cache.

The error logs can be maintained by choosing whether you want to keep old logs, remove logs after they reached a certain size or run a program to handle these logs.

Note The configuration file directives specified for the error logs apply to CGI error logs, as well.

To remove error log files, based on age, change the following configuration file directives.

ErrorLogArchive purge

ErrorLogExpire number-of-days

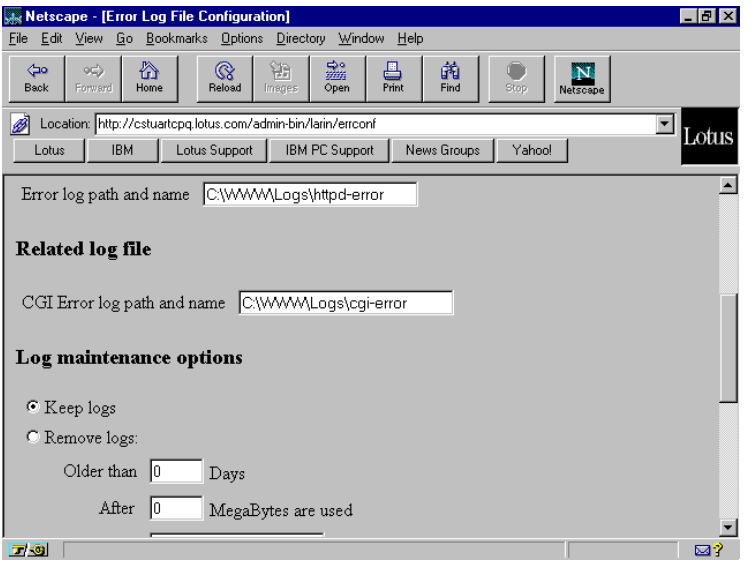
To remove error log files based on collective size, change the following configuration file directives.

ErrorLogArchive purge

ErrorLog **subdirectory**

SizeLimit number-of-megabytes

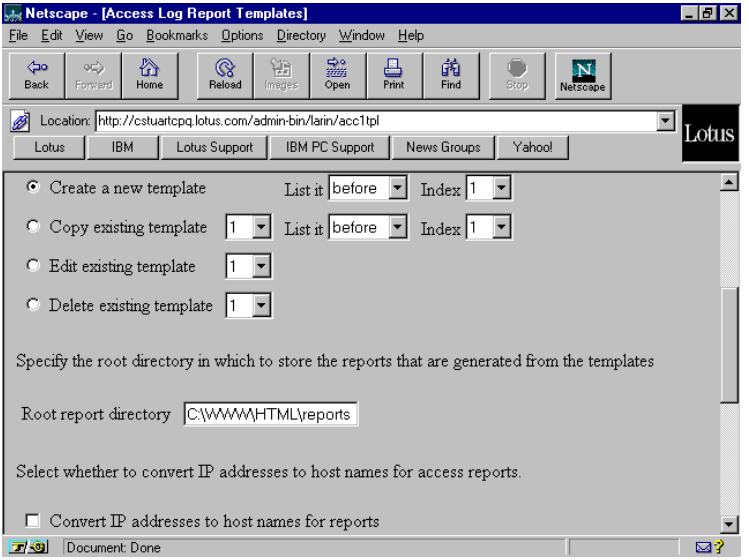
The following figure shows the error log configuration form showing the different fields where log maintenance options can be specified.



Creating Log Reports

The Domino Go Webserver creates reports that include some or all of the content of the access logs. At midnight each night, the server closes the current access log and creates a new access log file for the coming day. Reports are generated at that time using the access log that was just closed. Reports can also be generated for logs that have been archived. The reports default subdirectory is "\WWW\HTML\REPORTS".

Report generation of the Domino Go Webserver is a function of the "htlogrep" executable file located in the "\WWW\BIN" subdirectory. Before you can see a report, a report template must be created. The report template is stored as a report template file. The report template file is defined through the "Access Log Report Templates" configuration form or by directly editing the configuration file. The figure below is the "Access Log Report Templates" configuration form.



The example below creates a report template file that generates a report on the top 50 most frequently requested files and top 50 most frequent visitors. This example is coded directly into the configuration file.

AccessReportTemplate Top50 {

AccessReportDescription Top 50 most frequent visitors and accessed files

AccessReportTopList 50

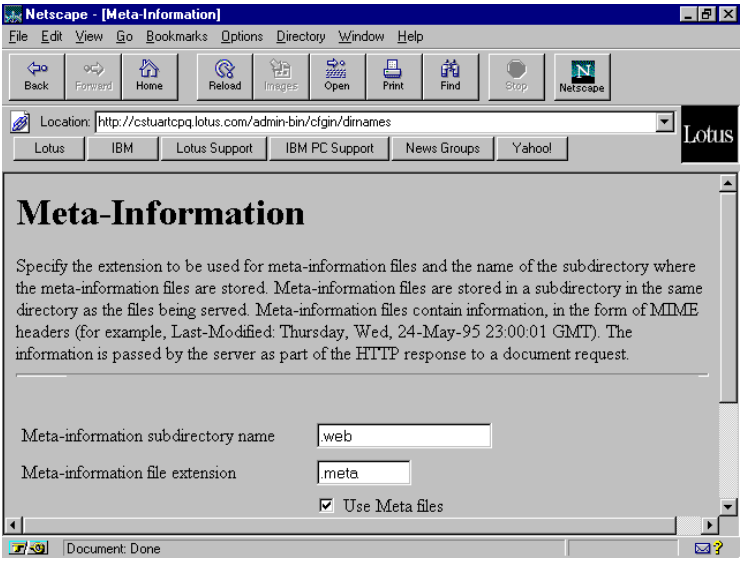
}

If you want to understand how your users navigate through the Web site, you can use Web usage mining statistics. These reports can identify where users enter and exit from a Web site and which Web pages are visited most. Identify user browsing patterns, which assist with better organization of Web pages. The reports are generated automatically and are not customizable except through the standard report templates.

Meta-Information

Meta-information describes the file containing a document, not the contents of the document. For example, meta-information for a file might give the date the file was created or modified. Any valid response headers like those described in the HTTP 1.1 specification can be included, as well as MIME headers. MIME header fields can include information, such as file type, subtype, encoding and content length.

The following figure shows the “Meta-Information” configuration form.



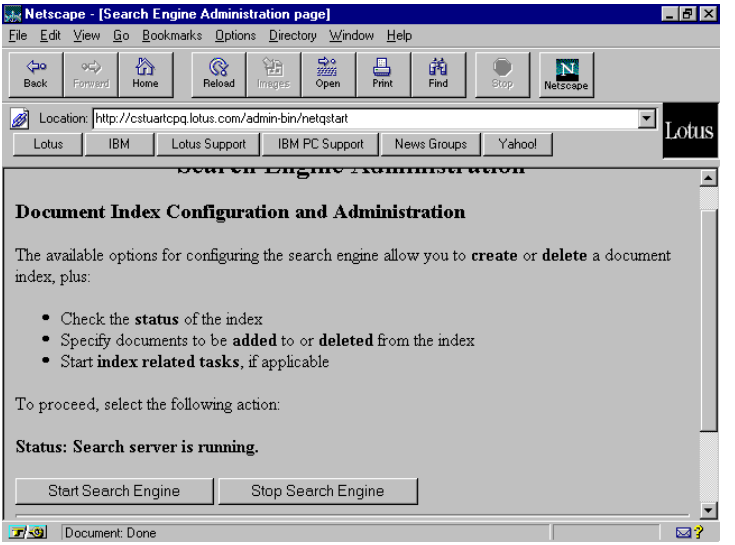
Note The dot character (.) at the beginning of the default value is used as part of the subdirectory name.

Search Engine

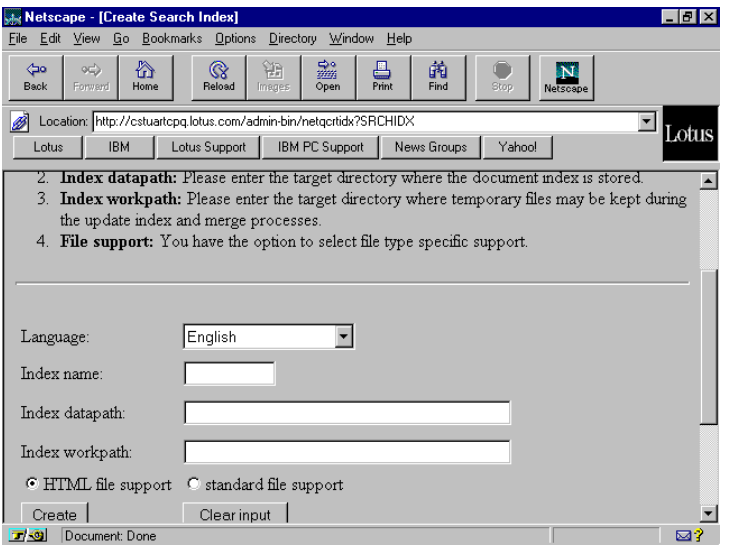
The Domino Go Webserver uses NetQuestion, which is a full-text search service that can be used as a global World Wide Web search service or as a centralized intranet search service. It is designed and optimized to handle the large amount of information that are typically stored in Web sites. NetQuestion supports the full spectrum of boolean search logic and free text search and also includes proximity search, phrase search, and front-middle-end masking of search terms. It offers sophisticated lexical affinities-based ranking for free text and hybrid queries (probabilistic search). NetQuestion can also detect misspellings in documents and expand search requests accordingly.

NetQuestion is specifically optimized for Web applications, in both intranet and Internet environments. It provides extremely fast indexing and retrieval where one precise index is built. Fuzzy searches can be done for English. For the other (single-byte character set) languages, precise searches are supported. NetQuestion features a compact index of about 35% to 40% of the document size. For customers not needing full context information (no need for proximity searches), an ultra-compact index of about 10% of the document size can be built.

The Domino Go Webserver search engine is started through the Configuration and Administration forms. The following figure shows the “Search Engine Administration” form; the search server status is “running.” The “start” and “stop” buttons are located near the bottom of the form.



Create a search index using the following “Search Engine Administration” form.

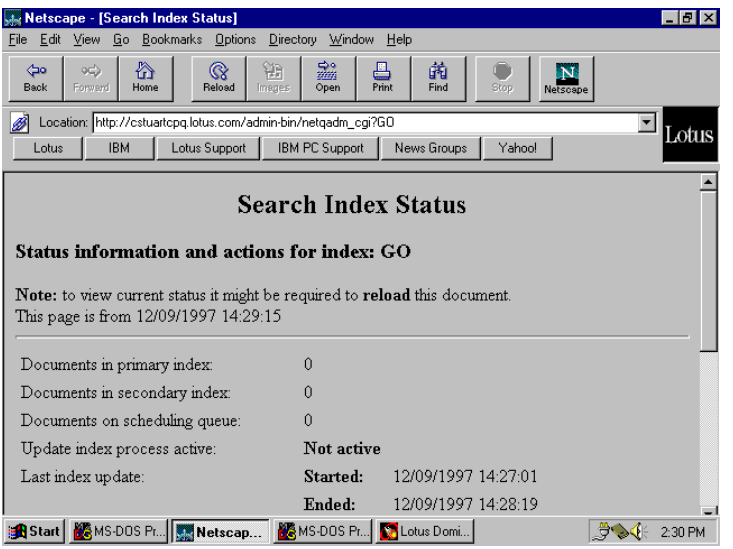


When NetQuestion builds indexes for your documents, it ignores the small inconsequential words (for example, articles and conjunctions) called stopwords. When creating an index, please ensure there is sufficient disk space available. Also, specify whether you want HTML or standard file support. The HTML file support drops all tags, and stores the document’s title within the document

index for later fast search result display. The standard file supports various other document formats. But, within the search result list, only the physical file is returned.

There is a form to view the status of an index. At the top of the form a table is displayed listing the items related to index document processing and the status of each of these items. The items listed include:

- Documents in primary index
- Documents in secondary index
- Documents in scheduling queue
- Update index process active
- Last index update
- Merge / Compress index active
- Search Process Active



System Management

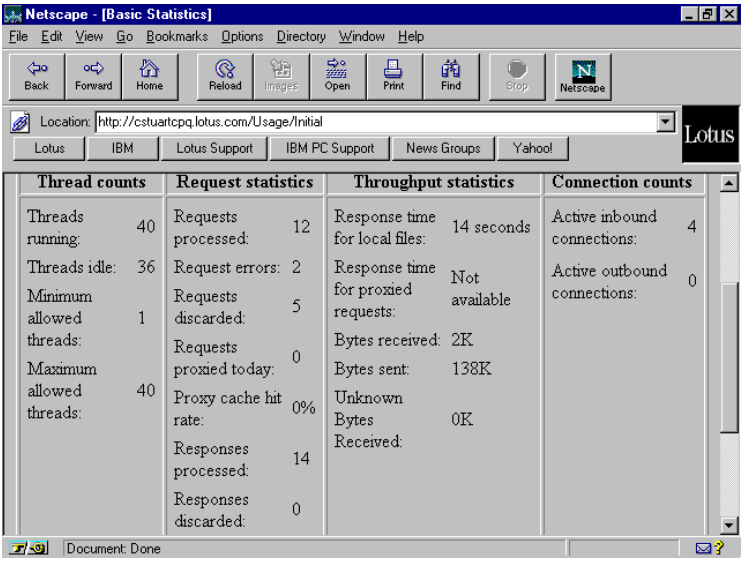
The Domino Go Webserver provides a set of forms to define SNMP server settings. These settings could be used to monitor and manage the health, throughput or the activity of your servers.

The “Server Activity Statistics” form displays server activity statistics. The server activity statistics include:

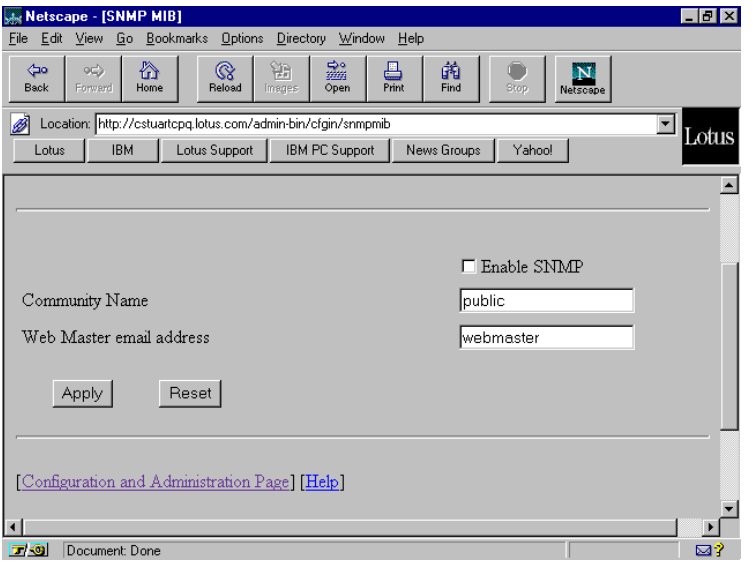
- Thread counts
- Request statistics

- Throughput statistics
- Connection counts

The server statistics can be used to determine if the number of threads are low, monitor server response by using the response time for local files, or monitor traffic by using the requests processed, bytes received, and bytes sent.



Another system management form is the “SNMP MIB” form:



If you would like to manage the server with an SNMP management agent, select to enable SNMP and specify the community name and the name of the system administrator who will manage the server.

Conclusion

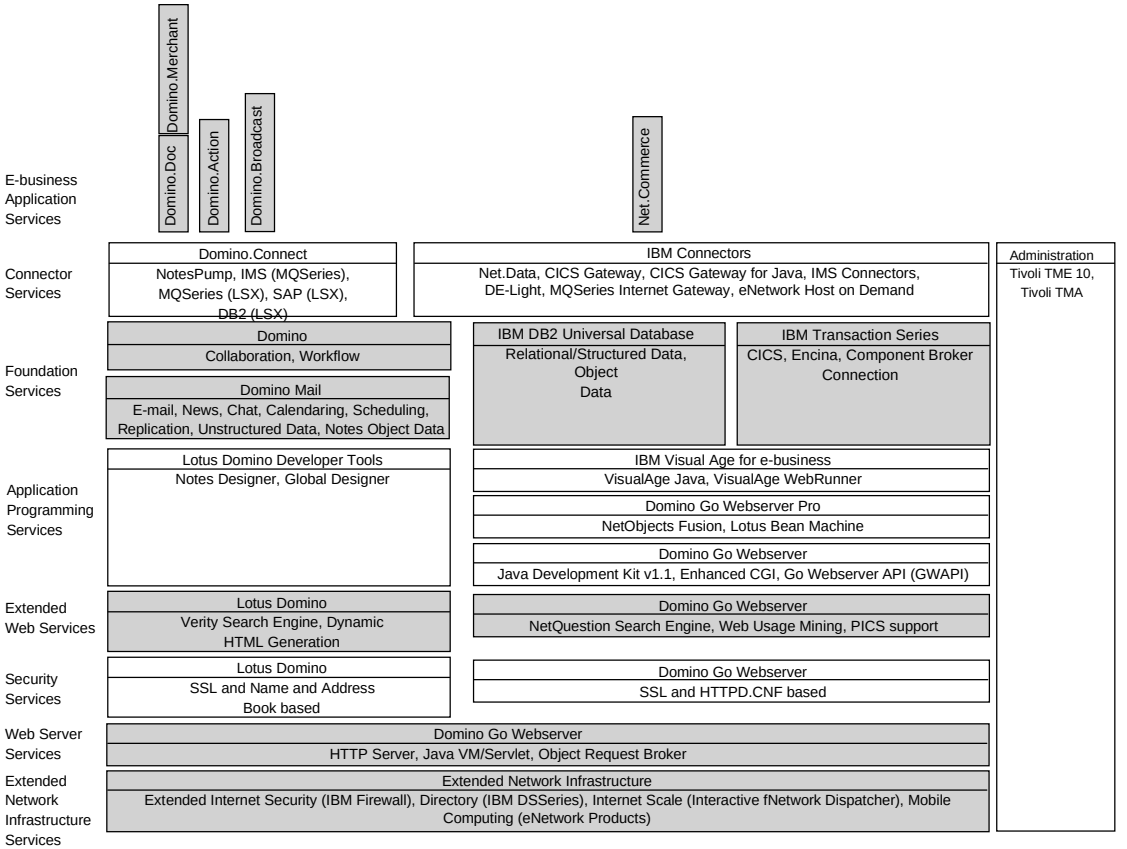
This chapter reviewed how to use the Domino Go Webserver administration and configuration forms to perform various system management tasks. Some of the issues regarding management of the Domino Go Webserver were also discussed.

Chapter 7
Product Evolution

This chapter describes how to evolve your Domino Go Webserver site through the NCF product line of servers. The NCF range of servers provide an evolution path that can support your Web site as it increases in complexity and size. The server products that support your Web site can evolve as well. We will describe the NCF Roadmap for product evolution and then discuss the individual product paths within the Roadmap.

NCF Product Evolution Roadmap

The NCF provides several product paths for evolving your Web site as it grows in functionality and size. These are referred to as Foundation Services in the NCF Product Roadmap section in Chapter 1: Architecture. These paths are not mutually exclusive but provide some common evolution scenarios based on common Web site characteristics. The following diagram describes the overall NCF product organization. This organization maps to the architecture described in Chapter 1:



The NCF defines four major server evolution paths from Domino Go Webserver. The evolution to these paths depends on the characteristics of your Web site. The following table summarizes some major Web site characteristics and a recommended product evolution for these sites.

Web Site Characteristic	*Recommended Server Product*	*Applicable Product Services*
Primarily used for standard communications	Domino Mail Server	E-mail, news, chat, calendaring, and scheduling
Primarily used to support or automate business processes	Domino Server	Collaboration and workflow
Primarily used as access to access structured or relational data	IBM DB2 Universal Database server	Relational Data
Primarily used to generate transactional information	IBM Transaction Series	Transaction

These servers have a wide range of function and scalablity. The functions include basic Web server activities, message-enabled workflow, and support for relational data and transactional application services.

These server products and the considerations for evolving to these products are discussed in the following sections.

Domino Go Webserver

Domino Go Webserver signifies an important addition to the Domino server family. It enables companies to deploy a solution with the assurance that they can leverage the complete Domino family. The entire range of Domino server products enable the development and deployment of advanced integrated messaging and collaborative applications for the Internet, extranets and intranets.

The Domino Go Webserver 4.6 release is now aligned with the Domino 4.6 product release. Domino Go Webserver is the next generation of the IBM Connection Secure Server (ICSS) product. New product enhancements will result in a much closer integration between future versions of Domino Go Webserver and Domino, resulting in a tightly integrated Domino Internet server family.

An important development consideration is that the Domino Go Webserver and Domino products share the same HTTP engine. This common core between the two products means that the Domino family provides a range of server products that can develop Web content ranging from static HTML file based to complex e-Business application services. The Lotus family of Internet servers provides the ability to start simple and grow fast as your Web business needs evolve.

There are two Domino Go products:

- Domino Go Webserver
- Domino Go Webserver Pro

The Domino Go Webserver Pro packages the NetObjects Fusion and Lotus BeanMachine with Go Webserver in a single offering. The NetObjects Fusion and Lotus BeanMachine are advanced Web site design and Java authoring tools.

Summary of Options for Evolving a Web Site

Lotus Domino Mail

Adds e-mail, news groups, chat, and calendaring and scheduling — all based on standard Internet protocols. Its goal is to extend the reach of a business's support to customers, partners and employees. It provides a more robust implementation of the directory and security services based on the Domino object store, while still adhering to the standard LDAP, X.509 v3, and SSL formats and protocols. With Domino Mail, developers have the option of exploiting the Domino document object store for managing their Web content.

Lotus Domino

Lotus Domino adds to Domino Mail by offering rich, mature workflow and collaboration solutions. The goal is to reduce product cycle times, support virtual teams, and enhance the efficiency of the enterprise. A prefabricated application framework (Domino Applications) and set of modules speeds the deployment of business applications. Domino.Action, Domino.Merchant, Domino.Doc, and Domino.Broadcast make it easy to deploy intranet, extranet and Internet business solutions that use the collaborative strengths of the Domino server. Domino.Applications provide integrated support for Web content management, commerce, and push technologies.

IBM DB2 Universal Database

DB2 is the industry's first scalable, Web enabled object-relational database, running on platforms ranging from Intel to UNIX systems, from uniprocessor to symmetric multi-processing (SMP) and massively parallel processing (MPP) environments. The IBM DB2 Universal Database handles tasks as diverse as assembling Web page elements, accessing distributed multimedia, data warehousing, powerful complex querying, and online transaction processing.

The IBM DB2 Universal Database offers a rich function set, ease of use, quality, security, and reliability combined with database tools, multimedia object-relational extenders and connectivity to the Web. Support in the IBM DB2 Universal Database for Net.Data, Java, and Java Database Connectivity (JDBC) provides Web integration that facilitates real-time e-Business applications.

IBM Transaction Series

IBM Transaction Series contains CICS and Encina connectors to integrate Web applications with enterprise systems. Evolving to IBM Transaction series adds CICS and Encina transaction monitor capabilities as well as

Component Broker Connector, which provides a rich object-oriented programming model and a full set of OMG object services. Transaction Series enables developers to develop, deploy, and administer exceptionally scalable, available, and secure applications for the Internet. Java facilities allow developers to build mission-critical server-side transactional applications that are called by applets.

Domino Go and Domino

Typically, Web sites evolve from simple static file-based solutions to full-function Web-enabled business solutions. Usually these fully Web-enabled business applications require:

- Messaging
- Collaboration
- Workflow
- Legacy Application Integration

Once the business requirements have been identified and the architecture has been agreed upon, it is a matter of selecting the best fit of technology available. Sometimes a combination of Domino Go Webserver and Domino products offers the best fit. In other cases, the best option could be to implement only Domino.

So what are the server alternatives? Can both Domino Go Webserver and Domino be part of the same server infrastructure? Or is it necessary to select one of the two products? How can earlier content development investments be protected? Can Domino Go Webserver content be moved unchanged to Domino? What are the functional similarities between Domino Go Webserver and Domino (Appendix F contains feature comparisons between Domino Go Webserver and Domino)? In the next section we will try to answer some of these questions.

What Are the Server Alternatives?

There are three main options to consider when deciding whether to use Domino Go Webserver, Domino or both. The Lotus Domino Go Webserver development team has successfully tested the following alternatives for Domino Go Webserver and Domino: The three alternatives are:

1. **Coexist**

   Domino Go Webserver and Domino concurrently running on the same physical machine

2. **Move**

   Domino Go Webserver and Domino implemented as part of the same server infrastructure, however running on physically separate machines

3. **Convert**

   Domino Go Webserver and Domino successfully hosting the same content type

   - HTML
   - Java servlets
   - CGI scripts

Which Alternative to Choose

Choosing between the three server alternatives depends upon the individual situation and the Web site requirements. The first task in determining an upgrade path to Domino is to evaluate some of the following criteria:

- **Hardware** Does the available hardware satisfy Domino's system requirements?
- **Content Complexity** Is there a substantial invest in HTML tools and processes? Do you want to continue this development strategy? How do you protect the already existing investment for content development?
- **Domino Plans** Is there a need to develop e-Business applications that benefit from services such as collaboration, or messaging?
- **Administration** Will the current administration team be responsible for Domino administration or is there a separate team?
- **Development** Will the current development team be responsible for Domino development or is there a separate team? What development tools are being used for content creation?

Upgrade Tools

Lotus is developing a tool to automate moving or converting Domino Go Webserver file based HTML content and CGI scripts to Domino. The tool is not yet available but it is planned to be released in three stages.

The first release of the tool will move HTML file-based content into a Domino .nsf database. Some of the tasks related with converting Domino Go Webserver content to Domino are:

- Move from the HTML file-based content into a Domino .nsf file format
- Move Java servlets, and CGI programs into directories accessible by Domino

The second phase will provide an automated utility to move file-based HTML content with embedded data objects, for example gifs, and CGI scripts to Domino. An objective of the tool is to assist with configuring Domino to secure and serve this content.

The third phase is a superset of the previous phases with the addition of a conversion capability between Domino Go Webserver and Domino's security models. This includes:

- Migrating page-level access control into Domino document-level access controls.
- Creating Domino Access Control Lists (ACL) roles to substitute for groups defined via the PROTECT directive statements.
- Migrating and defining users to the Domino public name and address book and optionally generating Domino user ids.
- Migrating certificates; this function is still under research.

First Server Coexistence Benefits

Domino Go Webserver and Domino can both participate as part of the same server infrastructure. It's recommended to run the server code for these products on physically different machines, particularly uniprocessor platforms, for two reasons:

1. Users may become dissatisfied with explicitly typing a TCP/IP port number as part of the URL request to access one of the two server nodes.
2. Lotus recommends Domino run on a dedicated application server.

Note There are massive parallel processor (MPP) hardware platforms, for example IBM SP2, that are asymmetrical, and permit each processor node to run its own operating system image and network connectivity. In this environment it would be possible to have a single physical machine running separate multiple instances of the server code per processor node.

The benefits associated with designing a solution combining both technologies would be:

- No need to move or convert Domino Go Webserver content to Domino.
- Content-specific application responsibilities can be assigned to each server. For example, Domino Go Webserver could manage all static HTML file based content while Domino hosts an e-Business application that requires messaging.
- URL links can be created that integrate content between the different servers and create the perception of a single logical Web site.
- HTML pages can be searched via the Net.Question search engine, which is only available on Domino Go Webserver (HTML file-search function is in plan for a future release of Domino).

Server Move Benefits

HTML, Java, and CGI program content developed for Domino Go Webserver can be moved without change to Domino. Essentially, you are using Domino simply as a HTTP server. The content gains no benefit from nor uses any of the available Domino advanced services unless you add it.

Another possibility is for Domino to concurrently host and serve the unchanged content from the Domino Go Webserver as well as from any Web-enabled Domino applications. This allows you to design new applications with Domino features that have a URL link to the file system content moved over from Domino Go Webserver. Over time the new development efforts can focus on delivering Domino Web-enabled application solutions, and minimizing the dependencies of file system-based content.

Additional benefits associated with moving Domino Go Webserver content to Domino include:

1. Centralized administration.
2. A single type of Web server to support.
3. Reduced server infrastructure costs by consolidating content onto a single server.
4. Migration strategy that permits the gradual reduction of file-based content.

Server Convert Benefits

Another consideration is to convert Domino Go Webserver file-based content to a Domino .nsf file format. The notion of content conversion may be misleading. The process recodes (or cuts and pastes) the existing logic from the HTML file into the Domino .nsf file. Currently, no tool is available to automate this effort; however, both IBM and Lotus are working to develop one.

Domino supports the coding of HTML source code and CGI variables directly into the .nsf file. Also, Domino can run any Java servlets and most CGI programs that were rewritten to the Domino Go Webserver GWAPIs to improve application performance (as recommended in Chapter 4: Performance). Technically, GWAPI programs can run with Domino; however, not all GWAPI functionality is supported for Domino.

There may be instances when all existing Domino Go Webserver file-based content is unable to be converted to Domino. Then a decision needs to be reached to invest in the effort to recode the logic within native Domino services.

The benefits of converting Domino Go Webserver content to Domino can be summarized as follows:

1. All content contained within Domino object store.
2. Access to Domino application services:
   - Field-level replication
   - Messaging
   - Collaboration
3. Single application development environment for content authoring.
4. Centralized administration.
5. Dynamically created HTML pages for browser access.

Skills Requirements

Each of the three server alternatives for evolving your Domino Go Webserver towards Domino requires different development and administrative skill sets. Evaluate existing skill sets to determine whether these skills will serve the new Web server architecture well.

Although Domino can host HTML, Java and CGI content, let's consider a hypothetical project plan that requires all Domino Go Webserver content to be moved unchanged to Domino. The project rationale is to first implement a single type of server infrastructure that can host any type of Web content, at minimum cost. Later on you can develop resources to convert the content to benefit from the various Domino services.

The content could be moved as is. What does that mean in respect to the people skills necessary to develop or administer the site? First, the development skills required to create the content remains constant since the content is unchanged. The development tools (Java, HTML, Perl) are simply moved onto the Domino platform.

Content developers tend to lean towards tools that they have mastered. The development skills acquired to develop Domino Go Webserver content are still valuable but as the development focus changes to Domino so does the need to learn to develop applications within the framework of Domino architecture to benefit from its many services. This is a slightly different development approach from that required for Domino Go Webserver, for example coding HTML directly into a Domino .nsf file.

Although, from a high level the Domino Go Webserver and Domino administrators appear to perform similar functions, the role differs in the tools and techniques to handle the daily operations of site administration. Most Domino administrators have legacy Lotus Notes experience, for

example, the Domino administrators are comfortable with the administration of the Domino public address book or the product .ini file. One of the integration strengths of both the Domino Go Webserver and Domino products is that there are some functional similarities such as ACLs, certificates, and authentication. However, the implementation of these functions varies between the two products. For example, managing the ACL is different. Again, the issue is bridging these two different worlds. There's a learning curve that administrators with legacy Lotus Notes experience need to climb to administer a Domino site that hosts unchanged Domino Go Webserver content.

The administration personnel issue needs consideration whether moving unchanged Domino Go Webserver content to Domino or implementing Domino Go Webserver and Domino as part of the same server infrastructure. When Domino Go Webserver content is converted to Domino the administration skill set acquired from legacy Notes experience becomes less of an issue since the content is simply contained within .nsf files.

Example of Server Upgrade Alternatives

This section details the experience reported with the testing and analysis of these server upgrade alternatives:

1. Coexisting
2. Moving
3. Converting

Initial Domino Go Webserver Configuration

First a site was created using Domino Go Webserver 4.6 installed on a Pentium 166 MHz machine with 64MB of RAM running NT and then, each upgrade alternative was assessed. The test site was based upon the hypothetical requirements of a bank and included typical banking functions. The three main tasks to create the test site were:

1. Creating content
2. Server configuration
3. Implementing security

1. Creating Content

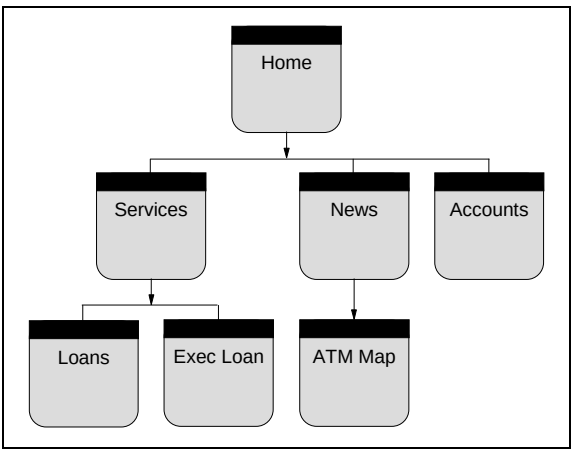
Content was developed using NetObjects Fusion 2.0.1. NetObjects Fusion is packaged with the Domino Go Webserver Pro. NetObjects Fusion is a development tool that assists with content design, development and maintenance.

Note For more detail information on a Web site creation using NetObjects Fusion, reference the "*Get Go Going*" report, on the IBM e-Business Web site:

`www.software.ibm.com/ebusiness/buzz.HTML`

There are other tools available for content authoring and the process would be consistent regardless of the tool selected. The Lotus BeanMachine was also used to generate a Java applet to add animation to the site.

The figure below shows the Web page layout as viewed from NetObjects Fusion:



The site was created with these Web page objects:

- The **Home** page contained animation and URL links to the other Web page objects. It also contained a Java applet that was created using BeanMachine and inserted via NetObjects Fusion.
- The **Services** page contained a table with URL links to the two loan Web pages.
- The **Loans** and **Exec Loans** Web pages contained forms that used Perl to process data and generate a Web page. (Perl was selected because it's a cross-platform scripting tool.) Each Web page is protected by a different security scheme.
- The **News** Web page contained a NetObjects Fusion "TickerTape" component and a URL link to the ATM Web page.
- The **ATM Map** is a Java applet that performed a search function to locate ATM machines.
- The **Accounts** Web page contained an object built with the Fusion drawing tool.

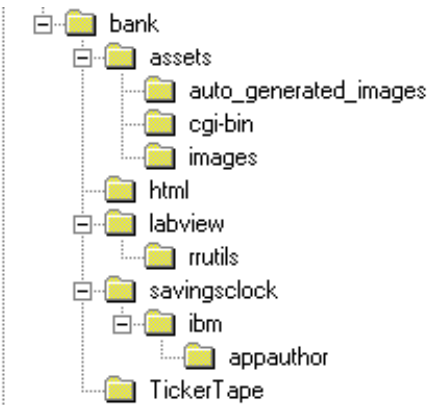
Using NetObjects Fusion both a staging area for developing and testing Web pages, and a production area for the publication of Web pages was created on the server. The staging and production areas are located at different points on the directory tree structure of the Web server. The staging and production areas, for the most part, contain identical file images of the content. All files were FTP'ed to the production area using Fusion. The production area directory path was:

```
D:\www\bankstage
```

The FTP process was repeated to move files to the staging area. The staging area directory path was:

```
D:\www\bankstage
```

The production file system directory tree structure looked as follows:



The **assets** subdirectory contained various image files. The **HTML** subdirectory contained the HTML text files. The **TickerTape** subdirectory contained a component that displayed a scrolling message. The **labview** and **savingsclock** subdirectories contained Java class files.

2. Server Configuration

First we defined a default home page for this site. This required modifying the server configuration file "httpd.cnf". The configuration file was located in the **D:\WINNT** subdirectory (NT default subdirectory). Updates to the "httpd.cnf" file can be done either through the Domino Go administration Web pages or by directly editing the file with a text editor.

Changes were made to the EXEC and PASS directives in the "httpd.cnf" configuration file:

```
EXEC /stage/HTML/bank/cgi-bin/*
D:\www\bankstage\assets\cgi-bin\*
```

Note This directive contains the staging location of the CGI Perl scripts.

```
EXEC /HTML/bank/cgi-bin/*        www\bank\assets\cgi-bin\*
```

Note This directive contains the production location of the CGI Perl scripts.

```
PASS /stage/*                    D:\www\bankstage\*
```

Note This directive contains the staging location of the HTML files.

```
PASS /go/*                       D:\www\HTML\*
```

Note This directive contains a link to the default Domino Go Webserver front page, which still needs to be accessed. This directive is necessary because the next directive specifies the bank's default home page.

```
PASS /*                          D:\www\bank\*
```

Note This directive contains the production location of the HTML files and also defines the default home page for the server.

The **EXEC** and **PASS** directives need to be coded within the "httpd.cnf" configuration file in this order to locate requested files. These directives are processed in the order specified in the configuration file.

3. Implementing Security

Next we created a protection method to permit public anonymous access to certain documents, as well as to create user IDs and group lists to restrict document access using the access control lists.

Group lists were created that contained members either by specifying a user name or by nesting another group name as part of the membership list. Then Web site areas were identified for the different levels of access.

The group **bank.group** was created using a text editor. For example, the group list was:

bank.group file

Group	*Members*
tellers	teller1, teller2
customers	customer1, customer2
executives	exec1, exec2
others	michele, mike, rudy, mario

The security scheme was set up as follows:

- Anyone in the password file, except executives can access the **Loans** Web page.
- Only executives can access the **Executive** loans Web page.
- Anyone can access the **ATM** Web page.
- Anyone can browse the **Home** Web page and public areas.

- Only the Webmaster can access the **staging** area.
- In this example, security for the administration function is not addressed.

User IDs and passwords were created using **htmadm.exe**, which is shipped with the Domino Go Webserver. The password file **bank.passwords** was located in the **D:\domprot** subdirectory (the subdirectory was created to store the security files).

The following **PROTECT** directives were added to the "httpd.cnf" configuration file. The **PROTECT** directives determine the type of access permitted for both the files and directories.

The URLs that require protection need to be listed, along with the protection method or the file that contains the protection method. Reference the Lotus *Domino Go Webserver Webmaster's Guide* for detailed information on the PROTECT directives.

The protect directives that were coded in the "httpd.cnf' file follow:

Protect Directive	*Path*	*Note*
PROTECT/STAGE/*	D:\domprot\bank.protectmichele	Staging area accessible only to Michele
PROTECT/HTML/bank/cgi-bin/loan.pl	D:\domprot\bank.protectloans	Regular loans accessible to tellers, customers and others
PROTECT/HTML/bank/cgi-bin/loan.pl	D:\domprot\banl.protectexec	Executive loans accessible to only executives
PROTECT/HTML/atm_map*	D:\domprot\bank.protectall	ATM accessible to anyone in password file

Each of the above directive statements coded in the "httpd.cnf" file points to another file that contains the protection method details. These files are located in the **D:\domprot** subdirectory. The protection method details could have been coded in the "httpd.cnf" configuration file directly; however, separate files were chosen because it's easier to maintain and it reduces the possibility of incorrectly coding the "httpd.cnf" file. In either case the format of the protection method is identical. Listed below are the contents of these files:

bank.protectmichele:

Parameter	*Value*
ServerID	Only_Michele
Authtype	Basic
PasswdFile	d:\domprot\bank.passwords
GroupFile	d:\domprot\bank.group
Getmask	michele
Mask	michele

bank.protectloans:

Parameter	*Value*
ServerID	Anyone_but_an_exec
Authtype	Basic
PasswdFile	d:\domprot\bank.passwords
GroupFile	d:\domprot\bank.group
Getmask	All
Postmark	tellers, customers, others
Mask	Anybody

bank.protectexecs:

Parameter	*Value*
ServerID	Just_for_execs
Authtype	Basic
PasswdFile	d:\domprot\bank.passwords
GroupFile	d:\domprot\bank.group
Getmask	All
Postmark	executives
Mask	Anybody

bank.protectall:

Parameter	*Value*
ServerID	All_in_the_group
Authtype	Basic
PasswdFile	d:\domprot\bank.passwords
GroupFile	d:\domprot\bank.group
Getmask	All
Mask	Anybody

Domino Server Installation Alternatives

First, Domino was installed on the same physical machine as the Domino Go Webserver. The other installation alternative for Domino and Domino Go Webserver was to install them on physically different machines. In both cases the minimum configuration used was a Pentium 166 MHz machine with 64 MB RAM running NT 4.0. Domino 4.6 was installed with the HTTP server task enabled. The HTTP task was added to the "SeverTasks" statement in the NOTES.INI file.

Note If both Domino and Domino Go Webserver are installed on the same physical machine, one of the servers needs to be configured for a different HTTP port, for example 8080. Also, a separate SSL port may be applicable, for example 8443. This can be done by modifying the "Port" and "SSL Port" fields in the HTTP section of the server document in the public address book.

Server Coexistence Alternative

Domino and Domino Go Webserver were implemented as part of the same server infrastructure with each installed on physically separate machines. The Domino Go Webserver continued to serve the file system-based content while any new development effort was focused on Domino. URL links were created between the documents physically located on different servers. There were no problems experienced with Domino and Domino Go Webserver coexisting either on the same physical machine or implemented separately but as part of the same server infrastructure.

Server Move Scenario

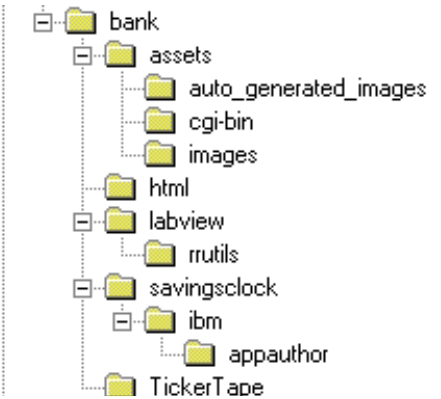
The Domino Go Webserver content was moved unchanged to Domino. There were five main tasks:

- Physically move the content to Domino.
- Configure Domino to serve the content.
- Implement security.
- Move the Domino Go Webserver administration tool to Domino.
- Test.

1. Move Content to Domino

If Domino and Domino Go Webserver are running on the same physical machine, then the content need not be moved because it is already there. However, if the server code is implemented on a physically separate machine, then it needs to be moved. In this case NetObjects Fusion was used to FTP the content to the Domino server. (There are other valid methods to move the content which may be more appropriate for your work.) To maintain a seamless server support environment, the directory

tree structure from Domino Go Webserver was recreated on Domino. The directory tree structure remained consistent between both servers after the move:



2. Server Configuration

This section will start with a brief explanation of Domino server configuration.

Domino Configuration Files

The Domino HTTP server task uses three sources for configuration at server initialization:

- Public Address Book (server document)
- Domino configuration database (domcfg.nsf)
- httpd.cnf configuration file

The Domino server document is located in the public name and address book. This document is used for some of the HTTP server configuration parameters, and it's recommended when possible that configuration changes be made through the server document. However, it does not include some of the parameters that were available with Domino Go Webserver, such as the PROTECT directives which limit access to parts of the server.

The "domcfg.nsf" database file is an additional configuration database that you can create in Domino (the file must be named "domcfg.nsf"). Its primary purpose is to permit administrators to configure URL mappings that allow Domino to translate the URL requests it receives to specific files on the server. The URL mappings specified in the "domcfg.nsf" database are actually translated to PASS and EXEC directives by the Domino server. There are some differences between using the "domcfg.nsf" database and the "httpd.cnf" file directly, such as specifying the order desired for the server to process PASS and EXEC directives, and the ability to provide user id / password security. Because of these differences and an attempt to keep the environment simple the "domcfg.nsf" file was not used.

The “httpd.cnf” file is the HTTP server configuration file that Domino uses for some of its configuration, in addition to the public name and address book and the “domcfg.nsf” database. The “httpd.cnf” configuration file is the same file for both the Domino and Domino Go Webserver. However, because Domino has the public name and address book, not every directive that was used by the Domino Go Webserver in the “httpd.cnf” file is used by Domino. For this reason the version of the “httpd.cnf” file shipped with Domino has been pared down.

When the Domino HTTP task is started, all three files are read. The order of processing is:

1. Public Address Book
2. domcfg.nsf database (if it exists)
3. httpd.cnf file

Domino then combines the information from these files into a virtual “httpd.cnf” file which is used for the duration of the HTTP server task. The order in which the HTTP server parameters are read is important to understand the behavior of Domino. For example, the PASS and EXEC directives are processed in the order which they are listed in the virtual “httpd.cnf” file. In the same way Domino Go Webserver depends on their order in the “httpd.cnf” file. When Domino receives an URL request, it searches the PASS and EXEC directives coded in the virtual “httpd.cnf” from the top until it finds a match. Once a match is found it discontinues the search even if a more appropriate match is found later in the list. This means that any PASS and EXEC directives in the public name and address book will be searched before those in the “domcfg.nsf” database, and the “httpd.cnf” file will be searched last.

Modifying the Domino “httpd.cnf” File

Changes to the “httpd.cnf” file have implications for the Domino server. The fact that it’s read after the public name and address book does not mean that it does not affect entries in the public name and address book. This is because certain directives can have multiple occurrences, and only the last one is used, while for others the first occurrence is used. As a general rule, follow these guidelines:

- If a directive can be implemented through the public name and address book, implement it there and do not modify the “httpd.cnf” file.
- If a directive cannot be implemented through the public name and address book, implement it through the “httpd.cnf” file. There is no Domino administrative interface for changing the ”httpd.cnf” configuration file. Always make a backup copy of the configuration file before changing it. Domino depends upon the file to load the HTTP server task, and if it is corrupted your HTTP server may not run properly.

- When coding PASS and EXEC directives for HTML files that are not stored in a Domino database determine the significance of the search order and security issues. If neither of these issues are important then use the "domcg.nsf" database. If either the search order or security are important then you must use the "httpd.cnf" file. This is because there's currently no way to specify the search order you want Domino to process the entries in the "domcfg.nsf" database, and the security of file system based HTML content can only be implemented through the "httpd.cnf" file.

How Domino Was Configured

The specific PASS and EXEC directives coded for the Domino Go Webserver were added to the Domino "httpd.cnf" file. A copy of the Domino "httpd.cnf" file was created before the file was changed.

These were the directives added to the Domino "httpd.cnf" file:

EXEC /stage/HTML/bank/cgi-bin/*
D:\www\bankstage\assets\cgi-bin*

EXEC /HTML/bank/cgi-bin/* **D:\www\bank\assets\cgi-bin***

PASS /stage/* **D:\www\bankstage**

PASS /icons/* **D:\notes\data\domino\icons***

PASS /* **D:\www\bank***

The directive "**PASS /icons/* D:\notes\data\domino\icons***" shown above was not needed for the Domino Go Webserver. This directive permits access to the icons that are shipped with Domino. This directive must be added to the "httpd.cnf" file because the PASS /* directive in the "httpd.cnf" file overrides the "icons directory" field in the public name and address book server document.

Also, the following PASS directive was omitted from the Domino "httpd.cnf" file:

PASS /go/* **D:\www\HTML***

The reason is it does not apply to the Domino server. The default home page for both the Domino Go Webserver and Domino remains consistent. However, there is still the need to access the front page shipped with the Domino Go Webserver. By specifying the above PASS directive the Domino Go Webserver permits accessing the front page by entering "go" on the browser URL location line. On Domino the site's default home page is coded in the "Default Page" field in the HTTP section of the server document in the public name and address book. If you need to access the originally shipped Domino front page type this browser URL request:

```
HTTP://server id/?Open
```

HTML and Domino Document Links

The steps below show how to link a NetObjects Fusion generated file-based HTML document to a Domino databases. Most of these steps would be applicable to any other HTML generating tool. For simplicity, a document from the Notes Help database was used to link with.

The following steps were used, but your approach may vary:

1. Use a browser (in this case Netscape Navigator 3.0), to retrieve the document to be linked:
   - Enter the URL **`HTTP://raleigh/?Open`** (returns a database listing for the Domino server).
   - Select the Notes Help database.
   - Open the "About this Database" documents twistee.
   - Open the "creating" twistee.
   - Select the "Tour: Opening the "About This Database" document.
2. Start another window, start NetObjects Fusion and open the Web site under development (bank.nod).
3. Use NetObjects Fusion to select the Web page in which the link is to be added.
4. Select a "text" object from the tool palette and create a text object that reads "Go see the latest Domino news".
5. Click on the text "Go see latest Domino news" to highlight it.
6. Click the right mouse button and select "Element Properties".
7. Select the "External Link" tab, since this link is external to the Fusion Web site.
8. In the Netscape window, use the mouse to highlight the URL in the location field, excluding the "HTTP / /:" part. The URL is:

   `raleigh/help4.nsf/bb950e9029a50754852563c000631dca/b2af91d3b70d4155852564d20015f5c3?OpenDocument`

9. With the location highlighted, press the crtl and insert keys simultaneously and copy the highlighted text to the clipboard.
10. Go to the Fusion window, after the URL field label, place the cursor on the entry field and press shift and insert to paste the URL into that field.
11. Click on the "Link" button and the process is complete. Once the Web site is published it will have a link to the Notes database.

Note Because Domino uses a very long string of characters to guarantee the uniqueness of each document, it is always best to use some type of cut and paste method when creating links between the HTML file system-based documents and Domino database documents. Any valid cut and paste method will work. If you are linking two Domino documents within a Domino database, then Domino handles it automatically.

3. Secure Access Control to the Site

To secure directories, add the PROTECT directives from the Domino Go Webserver "httpd.cnf" file to the Domino "httpd.cnf" file.

Note These PROTECT directives must be placed somewhere ahead of the EXEC and PASS directives.

The following directives were added to the "httpd.cnf" file:

PROTECT /stage/*
D:\domprot\bank.protectmichele

PROTECT /HTML/bank/cgi-bin/bank.pl
D:\domprot\bank.protectloans

PROTECT /HTML/bank/cgi-bin/execloan.pl
D:\domprot\bank.protectexec

PROTECT /HTML/atm_map* **D:\domprot\prtectall**

.

.

.

EXEC /...

PASS /...

In the case of the server alternative where Domino was installed on a physically separate machine, the files that are pointed to by each of these directives were copied into the **D:\domprot** subdirectory. The file names were:

bank.groups

bank.password

bank.protectmichele

bank.protectloans

bank.protectexecs

bank.protectall

4. Moving the Domino Go Administration Tool to Domino

To be able to add and delete users IDs for HTML page access control, move the **htadm.exe** tool to Domino. It will be copied from the Domino Go Webserver **D:\www\bin** subdirectory to the Domino **D:\notes\data** subdirectory.

5. Testing

The Domino server is now ready to serve the Domino Go Webserver content. The tests performed included:

- Access each Web page.
- Attempt to access a Web page using an invalid user id.
- Execute the CGI script by submitting a loan request.
- Execute Java applets through loan request process.

In all instances, there was no functional difference regarding content behavior between Domino Go Webserver and Domino and no problems were experienced during testing.

Conclusion

In summary, this chapter focused on an evolution path for a Web site. Web servers can scale from an entry-level site that hosts static documents created using HTML through enterprise e-Business sites that include a range of IBM and Lotus solutions.

Particular emphasis was placed upon the Web server alternatives regarding Domino Go Webserver and Domino. Issues regarding the upgrade path between Domino Go Webserver and Domino were considered. The intent was to demonstrate that the financial investment and development effort for content can be protected when upgrading from Domino Go Webserver to Domino.

Appendix A
Cryptographic Techniques

Notes and Internet security mechanisms make use of a number of common cryptographic techniques. It is important to have a good understanding of these techniques and, in general, we have assumed that you have some basic knowledge of them. However, this is a complex area, so in this section we present a brief overview of the important cryptographic techniques. We hope that you will find this a useful resource.

We will discuss five subject areas:

1. Symmetric key (or bulk) encryption
2. Public key encryption
3. Secure hash (or digest) functions
4. Digital signatures and other combinations of the above techniques
5. Certification mechanisms

If you want to learn more about cryptography, we recommend the RSA Frequently Asked Questions document at

`http://www.rsa.com/rsalabs/newfaq`

Symmetric Key Encryption

This is a grown-up version of the kind of secret code that most of us played with at some time during childhood. Usually these use a simple character replacement algorithm; if you want to encrypt a message, you just replace each letter of the alphabet with another. For example:

Original letter: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Replacement: GHIJKLMNOPQRSTUVWXYZABCDEF

In this case the letters in the alphabet have just been shifted seven places to the right, so HELLO WORLD would translate to NKRRU CUXRJ. The premise on which this code is based is that both the sender and the receiver know a common key, in this case the number of places to shift the letters. This *shared secret* allows the receiver of the message to reverse the encryption process and read the scrambled message. Symmetric encryption algorithms used by computers have the same elements as this simple example, namely a mechanism to scramble the message (also known as a cipher) and a shared secret (a key) that allows the receiver to unscramble the encrypted message.

The strength of a symmetric key cipher of this kind is dictated by a number of factors. For example, it is important that it effectively randomizes the input, so that two related clear-text messages do not produce similar encrypted results. Our childish example falls down badly in this area, because each letter always converts to the same encrypted result, and because it does not encrypt spaces. The kindergarten cryptanalyst can quite easily break the code by knowing that any one-letter word is likely to be an A. For full-strength symmetric ciphers, much of the work of the cryptanalyst involves trying to find patterns in the result of the algorithm, to use as a shortcut to breaking the code.

If the encryption algorithm has no flaws of this kind, the main factor governing its strength is the size of the *key space*; that is, the total number of possible shared secrets. Once again our simple example falls short, because it only has 25 possible keys. We could mount a *brute force* attack very easily by trying each key in turn until we found a message that makes sense. Real symmetric ciphers use numeric keys, usually of between 40 and 128 bits in size. Even for the smallest of these a brute force attack has to try, on average, 2 to the power 39 or about 550,000,000,000 possible keys. Each extra bit of key size doubles the key space.

Characteristics of Symmetric Key Algorithms

There are a number of symmetric key ciphers in use. We have listed the main ones below together with a brief description of their capabilities. However, they also share several common characteristics:

- They are fast and cause a relatively low system overhead. For this reason symmetric key encryption is often referred to as bulk encryption, because it is effective on large data volumes.
- The algorithms are published openly and there are no commercial licensing issues to be considered in implementing them.
- They all fall under the control of the U.S. National Security Agency export restrictions. The precise operation of these restrictions is not a simple matter, but in essence that means that:

  1. Any software incorporating cryptographic technology that is exported by a U.S. company has to have a special export license.
  2. If the product includes symmetric encryption code that can be used for encrypting an arbitrary data stream, the license will *only* allow unrestricted export *if* the key size is smaller than a given, NSA-specified, value.

  What this means is that to export full-strength cryptography, a company has to have a special license for each customer. Such licenses are only issued for customers that the U.S. government considers to be friendly, such as major banks and subsidiaries of U.S. companies. Until recently

the threshold key size for a general export license was 40 bits. Several challenges have shown that a brute force attack can be mounted against a 40-bit key with relatively modest computing power. A government announcement in October 1996 opened the door to the use of larger keys, initially up to 56 bits, with the promise of unlimited key sizes when the computer industry develops effective key recovery technology. (Key recovery means that the key for a session can be discovered, given the knowledge of some other, master, key). 56 bits may not sound a lot better than 40, but in fact it is 2 to the power 16, or 65,536 times more difficult to crack.

Common Symmetric Key Algorithms

DES

The Data Encryption Standard is the most commonly used bulk cipher. It was originally developed by IBM in 1977 and it has resisted all attempts at cryptanalysis. DES breaks the data to be encrypted into a sequence of 64-bit blocks and then uses a 56-bit key to apply a sequence of mathematical transforms to it. There are a number of variations on the standard DES algorithm, such as cipher block chaining, in which each block of data is XOR'd with the previous block before encryption, and triple-DES in which DES is applied three times in succession.

RC2/RC4

These related ciphers were developed by RSA Data Security, Inc. RC2 is a block cipher similar to DES, whereas RC4 operates on a stream of data. They both use a 128-bit key, but they support key masking. This means that part of the key may be set to a known value so that the effective key size is whatever remains from the 128-bit total. This has been used to good effect in producing 40-bit versions of software products for export.

IDEA

The International Data Encryption Algorithm is another block cipher, in the mold of DES. It uses a 64-bit block size and a 128-bit key. IDEA is the bulk encryption algorithm used by Pretty Good Privacy (PGP).

Public Key Encryption

A non-mathematician can intuitively understand how a symmetric key cipher works by extrapolating from a familiar base. However, public key mechanisms are much less accessible to the lay person. In fact, it sometimes seems more like magic than technology. The fundamentals of public key encryption are:

1. Instead of a single encryption key, there are two related keys, a *key pair*.
2. Anything encrypted using one of the two keys can *only* be decrypted with the other key of the pair.

Let us say that two people, Bob and Alice, want to exchange messages using a public key algorithm. Alice generates a key pair and places one of the keys, the *private key* in a safe place. She sends the other half of the key pair (called, naturally, the public key) to Bob. Bob can now encrypt a message using the public key and only the owner of the matching private key, Alice can decrypt it. Of course, if Alice wants to send a reply, Bob needs to create his own key pair and send the public key to Alice. The big advantage of this mechanism over the symmetric key mechanism is that there is no longer any secret to share. In fact, it does not matter who has the public key, because it is useless without the matching private key.

There is one further advantage that public key gives us. In the above example, imagine that Alice uses her private key to encrypt a message and sends it to Bob. The message that is sent between them is still scrambled, but it is no longer private, because anyone with the public key can decrypt it (and we have said that we do not care who has the public key). So, what *can* we use this message from Alice for? The answer is: authentication. Because only Alice has access to the private key that created the message, it can *only* have come from her.

Characteristics of Public Key Encryption

Public key algorithms can trace their ancestry back to the Diffie-Hellman key exchange mechanism. Diffie-Hellman is not a general purpose encryption scheme, but rather a method of exchanging a secret key. There is only one widely-used general-purpose public key mechanism, the Rivest, Shamir and Adelman (RSA) algorithm, which is the property of RSA Data Security, Inc. Public key, like any encryption mechanism, relies on the fact that certain mathematical problems are very hard to solve. The problem that the RSA algorithm relies on is the factorization of large numbers. For a detailed description of the mathematics behind RSA public key encryption, refer to

`http://www.rsa.com/rsalabs/newfaq/q8.html`

Clearly, public key has a big practical advantage over symmetric key, in that there is no need to securely share a secret between the sender and receiver. However, the RSA algorithm is a much less efficient encryption technique than any commercial symmetric key algorithm, by a factor of 100 or more. It is therefore not a good choice for encryption of bulk data.

RSA is subject to the same US export restrictions as symmetric algorithms. However, the key in this case is actually an enormous number. Very roughly, an RSA key size of 1024 bits corresponds to a full-strength symmetric key of 64 bits or more.

Secure Hash Functions

The third tool in our encryption armory is not actually an encryption mechanism at all. A secure hash is a way of creating a kind of fingerprint of a message. A secure hash function has three main attributes:

1. It takes a message of any size and generates a small, fixed size, block of data from it (called a *message digest*). Re-executing the hash function on the same source data will always yield the same resulting digest.
2. It is not predictable in operation. That is to say, a small change in the source message will have an unpredictably large effect on the final digest.
3. It is, for all intents and purposes, irreversible. In other words there is no way to derive the source data, given its digested form.

What use is a secure hash function? Its main function is to detect whether a piece of data has been modified or not. We see in :hdref refid=crycomb. how this is used in combination with RSA to generate a digital signature.

There are two secure hash algorithms in common use. The most widely-implemented is MD5, which was developed by RSA Data Security, Inc. This generates a 128-bit digest from any length of input data. It is described in RFC1321. The other algorithm that is becoming increasingly common is the U.S. government-developed Secure Hash Standard (SHS). This produces a 160-bit digest, slightly larger than MD5.

Combinations of Cryptographic Techniques

Although the security protocols of Notes and the World Wide Web differ in detail they both use the three techniques above. Two combinations of the techniques are used very commonly, so we outline them here:

- **Public key delivery of a symmetric key**

  It is most efficient to use a symmetric key algorithm for bulk data, but first you have to copy the key from sender to receiver. A common approach is to use a public key algorithm to protect the key transfer process.

- **Digital Signatures**

  You do not always want to encrypt data in transit. Very often the contents of a message may not be secret, but you do want to be sure that it really came from the apparent sender. If Alice wants to prove to Bob that it was really she who sent him a message, she will attach a signature to the message exactly as she would if writing a real letter on paper. In the digital case she creates the signature by first generating a digest of the

message and then encrypting the digest with her private key. When Bob receives the message he first decrypts the digest (with Alice's public key) and then generates his own digest from the received message. If the two digests match, Bob knows that:

1. The message is the same as when it was sent (because the digest is the same).
2. It really was sent by Alice, because only she has the private key.

Public Key Certificates

We have seen how public key cryptography overcomes the problem of having to pass a secret from sender to receiver. There is still a need to send a key, but now it is a public key that anyone can see because it is only useful to an attacker if he also has the private key. However, this overlooks one crucial element of trust: how can you be sure that the public key really came from who you think it came from?

One answer is to only pass public keys to someone you know. Let's assume that Bob and Alice have known each other for a long time, so they could share their public keys by exchanging diskettes. For normal cases, however, you need some way to be sure that a public key is authentic.

The mechanism for doing this is the *public key certificate*. This is a data structure containing a public key, plus details of the owner of the key, all digitally signed by some trusted third party. Now when Alice wants to send Bob her public key she actually sends a certificate. Bob receives the certificate and checks the signature. As long as it has been signed by a certifier that he trusts, he can accept that this really is Alice's key.

Certificates in real life are more complex than this. Descriptions of how they are used in a variety of ways in the detailed sections about SSL and Notes security in the redbook: The Domino Defense: Security in Lotus Notes and the Internet, SG24-4848-01.

Appendix B
GWAPI Example

As we discussed in Chapter 6: Administration, the Domino Go Webserver logging functions combines all the log items together. This program, using the Go Webserver APIs, provides a "splitlog" function that creates multiple logs and writes to them in a multiple virtual host environment.

The program cuts the domain name from the Web server names and uses the result as the subdirectory name in the Logs directory. In the example that we use in Chapter 6, the Web server names were GOWEBx.LOTUS.COM, so cutting LOTUS.COM from that we have GOWEB1, GOWEB2 and GOWEB3 and the subdirectory for the logs would be: c:\www\Logs\goweb1 and so on. One subdirectory is used for each Web server.

Caution In a couple of instances in the code that follows, a line will spill over onto 2 lines. Please check the syntax carefully if you intend to use this program.

```
/************************************************************************
*  File:          splitlog.c
*
*  Example:
*    Configuration file (httpd.conf) Directives:
*      OS/390
*           Log   /*   /web/bin/splitlog.o:log
*
*      UNIX:
*           Log   /*  /usr/internet/server_root/cgi-bin/splitlogs.o:log
*
*      OS/2, Windows NT and Windows 95:
*           Log   /*   c:\www\cgi-bin\splitlogs:log
*
*  Browser Request:
*  The function log will be invoked for all url requests.
*
************************************************************************/
#pragma export(log)

#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <string.h>
#include <time.h>
#ifdef _OE_
#include <time.h>
#endif
#ifdef HPUX
#include <sys/time.h>
#endif
#include "HTAPI.h"
```

```
char * time_escapes (char * filename);
char * get_time (void);


/*
 *  Create and/or open text files for logging and log information to
 *  appropriate files
 */


void
HTTPD_LINKAGE
log (
    unsigned char *handle,
    long *return_code)
{
    unsigned char value[1000];
    char *client_addr, *remote_ident, *remote_user;
    char *request_method, *path, *server_prot, *time_stamp, *response;
    char *content_length, *user_agent, *referer;
    unsigned long name_len, value_len=sizeof(value);
    long rc;
    int i, alpha = 0;
    FILE *fp_access, *fp_agent, *fp_referer, *fp_err;
    unsigned long server_len, msg_len;
    char server[1000], path2[1000], err_msg[1000].,filename[1000];
    char *p, *name;


    fp_access = NULL; fp_agent = NULL; fp_referer = NULL; fp_err = NULL;

    server[0] = '\0'; server_len = sizeof(server); name_len =11;

    /* * * * extract value from http request * * * */
    HTTPD_extract(NULL,"SERVER_NAME", &name_len, server, &server_len, &rc);
    if (HTTPD_SUCCESS == rc)
    {
        server[server_len] = '\0';
    }
    else
    {
        /* sprintf(err_msg,"SplitLog:\tHTTPD_extract failed(%ld) for
        SERVER_NAME\n",rc);
           fprintf(stderr,"SplitLog:\tHTTPD_extract failed(%ld) for
           SERVER_NAME\n",rc);
           HTTPD_log_error(handle, err_msg, &msg_len,&rc);
        */
        *return_code = HTTP_NOACTION;
        return;
    }

/*
 *  Parse SERVER_NAME for long URLs.  Log directory is based on this
 *  parsed name.
 */

    for(i = 0; i < strlen(server);i++)
    {
         if(isalpha(server[i]) && (server[i] != '.'))
         {
               alpha++;
               break;
         }
    }

    if(alpha)
    {
```

```
     for(i = 0; i < strlen(server);i++)
     {
        if(server[i] == '.')
         {
   alpha++;
   if(3 == alpha)
   {
            server[i] = '\0';
          break;
   }
         }
      }
     }
     fprintf(stderr,"+++Splitlog: parsing SERVER_NAME as \"%s\" \n",server);
/*
 *  End parsing of SERVER_NAME.
 */

/*
 *  Change path2 below to path to contain logging information.
 */

    strcpy(path2,":hp2./www/logs/:ehp2.");
    strcpy(filename,"/errors.");
    name = time_escapes(filename);

    p = malloc(strlen(path2) + strlen(server) + strlen(name) + (2*sizeof(char)));
    if (NULL != p)
    {
        strcpy(p, path2);strcat(p, server);strcat(p,name);
#if 0
 fprintf(stderr, "p = %s\n", p);
#endif
        if (NULL != (fp_err = fopen(p,"a")))
            setbuf (fp_err, NULL);
        else
        {

            sprintf(err_msg,"SplitLog:\tError opening file %s \n",p);
            msg_len = strlen(err_msg);
#if 0
            fprintf(stderr,"COULDN't OPEN ERROR FILE\n");
#endif
            HTTPD_log_error(handle, err_msg, &msg_len, &rc);
            free(p);
            free(name);
            *return_code = HTTP_NOACTION;
            return;
        }
    }
    else
    {
        sprintf(err_msg,"SplitLog:\tMemory alloc failure for file %s\n", p);
        msg_len = strlen(err_msg);
        HTTPD_log_error(handle, err_msg, &msg_len, &rc);
    }

    free(p);
    free(name);

    strcpy(filename,"/access.");
    name = time_escapes(filename);

    p = malloc(strlen(path2) + strlen(server) + strlen(name) + (2*sizeof(char)));
    if (NULL != p)
    {
        strcpy(p, path2);strcat(p, server);strcat(p,name);
```

```
    if (NULL != (fp_access = fopen(p,"a")))
        setbuf (fp_access, NULL);
    else
        (void)fprintf(fp_err,"SplitLog:\tError opening file %s \n",p);
}
else
{
    (void)fprintf(fp_err,"SplitLog:\tMemory alloc failure for file %s\n", p);
}

free(p);
free(name);
strcpy(filename,"/agent_log.");
name = time_escapes(filename);

 p = malloc(strlen(path2) + strlen(server) + strlen(name) + (2*sizeof(char)));
if (NULL != p)
{
    strcpy(p, path2);strcat(p, server);strcat(p, name);
    if (NULL != (fp_agent = fopen(p,"a")))
        setbuf (fp_agent, NULL);
    else
        (void)fprintf(fp_err,"SplitLog:\tError opening file %s \n",p);
}
else
{
    (void)fprintf(fp_err,"SplitLog:\tMemory alloc failure for file %s\n", p);
}

free(p);
free(name);

strcpy(filename,"/referer_log.");
name = time_escapes(filename);

 p = malloc(strlen(path2) + strlen(server) + strlen(name) + (2*sizeof(char)));
if (NULL != p)
{
    strcpy(p, path2);strcat(p, server);strcat(p, name);
    if (NULL != (fp_referer = fopen(p,"a")))
        setbuf (fp_referer, NULL);
    else
        (void)fprintf(fp_err,"SplitLog:\tError opening file %s \n",p);
}
else
{
    (void)fprintf(fp_err,"SplitLog:\tMemory alloc failure for file %s\n", p);
}

free(p);
free(name);


client_addr = NULL;
remote_ident = NULL;
remote_user = NULL;
request_method = NULL;
path = NULL;
server_prot = NULL;
time_stamp = NULL;
response = NULL;
content_length = NULL;
user_agent = NULL;
referer = NULL;

name_len= 11;
HTTPD_extract (NULL, "REMOTE_HOST", &name_len, value, &value_len, &rc);
```

```
    if (HTTPD_SUCCESS != rc)
    {
       /*  (void)fprintf(fp_err,"SplitLog \tHTTPD_extract failed(%ld)
for REMOTE_HOST, trying to extract REMOTE_ADDR\n",rc); */
     name_len= 11;
        value_len = sizeof(value);
     HTTPD_extract (NULL, "REMOTE_ADDR", &name_len, value, &value_len, &rc);
     if (HTTPD_SUCCESS != rc)
     {
        /* (void)fprintf(fp_err,"SplitLog:\tHTTPD_extract failed(%ld)
for REMOTE_ADDR\n",rc); */
 }
 else
     {
         value[value_len] = '\0';
         client_addr = malloc(strlen((const char*)value) + 1);
         strcpy(client_addr, (const char*)value);
 }
    }
    else
    {
        value[value_len] = '\0';
        client_addr = malloc(strlen((const char*)value) + 1);
        strcpy(client_addr, (const char*)value);
    }

    name_len=12;
    value_len = sizeof(value);   /* Gets set in the previous extract.
                         Has to be set again to 1000 */
    HTTPD_extract (NULL, "REMOTE_IDENT", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
        /* (void)fprintf(fp_err,"SplitLog:\tHTTPD_extract failed(%ld)
for REMOTE_IDENT\n",rc); */
 remote_ident = malloc(strlen("-") + 1);
        strcpy(remote_ident, "-");

    } /* endif */
    else
    {
        value[value_len] = '\0';
        remote_ident = malloc(strlen((const char*)value) + 1);
        strcpy(remote_ident, (const char*)value);
    }

    name_len=11;
    value_len = sizeof(value);   /* Gets set in the previous extract.
                         Has to be set again to 1000 */
    HTTPD_extract (NULL, "REMOTE_USER", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
        /* (void)fprintf(fp_err,"SplitLog:\tHTTPD_extract failed(%ld)
for REMOTE_USER\n",rc); */
 remote_user = malloc(strlen("-") + 1);
        strcpy(remote_user, "-");
    } /* endif */
    else
    {
        value[value_len] = '\0';
        remote_user = malloc(strlen((const char*)value) + 1);
        strcpy(remote_user, (const char*)value);
    }


    name_len=14;
    value_len = sizeof(value);   /* Gets set in the previous extract.
                         Has to be set again to 1000 */
```

```
    HTTPD_extract (NULL, "REQUEST_METHOD", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
        /* (void)fprintf(fp_err,"SplitLog:\tHTTPD_extract failed(%ld)
for REQUEST_METHOD\n",rc); */
    } /* endif */
    else
    {
        value[value_len] = '\0';
        request_method = malloc(strlen((const char*)value) + 1);
        strcpy(request_method, (const char*)value);
    }

    name_len=5;
    value_len = sizeof(value);   /* Gets set in the previous extract.
                        Has to be set again to 1000 */
    HTTPD_extract (NULL, "PPATH", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
        /* (void)fprintf(fp_err,"SplitLog:\tHTTPD_extract failed(%ld)
for PATH\n",rc); */
    } /* endif */
    else
    {
        value[value_len] = '\0';
        path = malloc(strlen((const char*)value) + 1);
        strcpy(path, (const char*)value);
    }

    name_len=15;
    value_len = sizeof(value);   /* Gets set in the previous extract.
                        Has to be set again to 1000 */
    HTTPD_extract (NULL, "SERVER_PROTOCOL", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
        /* (void)fprintf(fp_err,"SplitLog:\tHTTPD_extract failed(%ld)
for SERVER_PROTOCOL\n",rc); */
        *return_code = HTTP_SERVER_ERROR;
    } /* endif */
    else
    {
        value[value_len] = '\0';
        server_prot = malloc(strlen((const char*)value) + 1);
        strcpy(server_prot, (const char*)value);
    }



    name_len=13;
    value_len = sizeof(value);   /* Gets set in the previous extract.
                        Has to be set again to 1000 */
    HTTPD_extract (NULL, "HTTP_RESPONSE", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
        /* (void)fprintf(fp_err,"SplitLog:\tHTTPD_extract failed(%ld)
for RESPONSE\n",rc); */
    } /* endif */
    else
    {
        value[value_len] = '\0';
        response = malloc(strlen((const char*)value) + 1);
        strcpy(response, (const char*)value);
    }

    name_len=23;
    value_len = sizeof(value);   /* Gets set in the previous extract.
                        Has to be set again to 1000 */
```

```
    HTTPD_extract (NULL, "RESPONSE_CONTENT_LENGTH", &name_len, value, &value_len,
&rc);
    if (HTTPD_SUCCESS != rc)
    {
        content_length = malloc(3);
        strcpy(content_length, "0");
    } /* endif */
    else
    {
        value[value_len] = '\0';
        content_length = malloc(strlen((const char*)value) + 1);
        strcpy(content_length, (const char*)value);
    }

    name_len=15;
    value_len = sizeof(value);
    HTTPD_extract (NULL, "HTTP_USER_AGENT", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
    }
    else
    {
        value[value_len] = '\0';
        user_agent = malloc(strlen((const char*)value) + 1);
        strcpy(user_agent, (const char*)value);
    }

    name_len=12;
    value_len = sizeof(value);
    HTTPD_extract (NULL, "HTTP_REFERER", &name_len, value, &value_len, &rc);
    if (HTTPD_SUCCESS != rc)
    {
    }
    else
    {
        value[value_len] = '\0';
        referer = malloc(strlen((const char*)value) + 1);
        strcpy(referer, (const char*)value);
    }


    time_stamp = get_time();

    if (fp_access != NULL)
    (void)fprintf(fp_access,"%s %s %s [%s] \"%s %s %s\" %s %s\n",
                client_addr,
                remote_ident,
                remote_user,
                time_stamp,
                request_method,
                path,
                server_prot,
                response,
                content_length);

    if (fp_agent != NULL)
    (void)fprintf(fp_agent,"[%s] \"%s\"\n",
                time_stamp,
                user_agent );

    if (fp_referer != NULL)
    (void)fprintf(fp_referer,"[%s] \"%s\"\n",
                time_stamp,
                referer );

    if(client_addr)     free(client_addr);
    if(remote_ident)    free(remote_ident);
```

```
    if(remote_user)     free(remote_user);
    if(time_stamp)      free(time_stamp);
    if(request_method)  free(request_method);
    if(path)            free(path);
    if(server_prot)     free(server_prot);
    if(response)        free(response);
    if(content_length)  free(content_length);
    if(user_agent)      free(user_agent);
    if(referer)         free(referer);


    (void)fclose(fp_access);
    (void)fclose(fp_agent);
    (void)fclose(fp_referer);
    (void)fclose(fp_err);

    *return_code = HTTP_NOACTION;

}


#define DATE_EXT_LEN 21
#define HTTP_TIME_LEN 43

#define HTABS(x)  (((x) < 0) ? (-(x)) : (x))

char * time_escapes (char * filename)
{
    time_t t;
#ifndef _OE_
    struct tm Result;
#endif
    struct tm * gmt;
    char * result;
    char date[DATE_EXT_LEN];
    char *ptr;
#ifdef HPUX
    int iretval;
#endif

    time(&t);

#if defined(HPUX)

    iretval = localtime_r(&t, &Result);
    gmt = &Result;
#elif defined(_OE_)
    gmt = localtime(&t);
#else
    gmt = localtime_r(&t, &Result);
#endif

    result = malloc(strlen(filename)+DATE_EXT_LEN+2);

    if (gmt)
    {
        strftime(date, DATE_EXT_LEN, "%y%m%d", gmt);
      *(date+DATE_EXT_LEN) = '\0';
    }
    else
      *date = '\0';

    ptr = date;
    while (*ptr)
    {
        if (*ptr == ' ')
            *ptr = '_';
```

```
        ptr++;
    }

    if (*(filename+strlen(filename)-1) == '.')
    {
        sprintf(result, "%s%s", filename, date);
    }
    else
    {
        sprintf(result, "%s.%s", filename, date);
    }

    return result;
}


char * get_time (void)
{
    time_t t;
#ifndef _OE_
    struct tm Result;
#endif
    struct tm * gmt;
    char * result;
    char date[HTTP_TIME_LEN+1];
    char *ptr;
#ifdef HPUX
    int iretval;
#endif
    char tzString[32];
    int z;

    time(&t);

#if defined(HPUX)

      iretval = gmtime_r(&t, &Result);
      gmt = &Result;
#elif defined(_OE_)
      gmt = gmtime(&t);
#else
      gmt = gmtime_r(&t, &Result);
#endif

    result = malloc(HTTP_TIME_LEN+2);

    if (gmt)
    {
        strftime(date, DATE_EXT_LEN, "%d/%b/%Y:%H:%M:%S ", gmt);
      *(date+DATE_EXT_LEN) = '\0';
    }
    else
      *date = '\0';
/*
    z = timezone/60;
    sprintf(tzString, " %c%02d%02d",z<0?'-' :'+', (int) HTABS(z)/60,
                                                  (int) HTABS(z)%60);
*/
    strcpy(result,date);
    strcat(result,tzString);
    return result;

}

/* EOF */
```

Appendix C
Net.Data Example

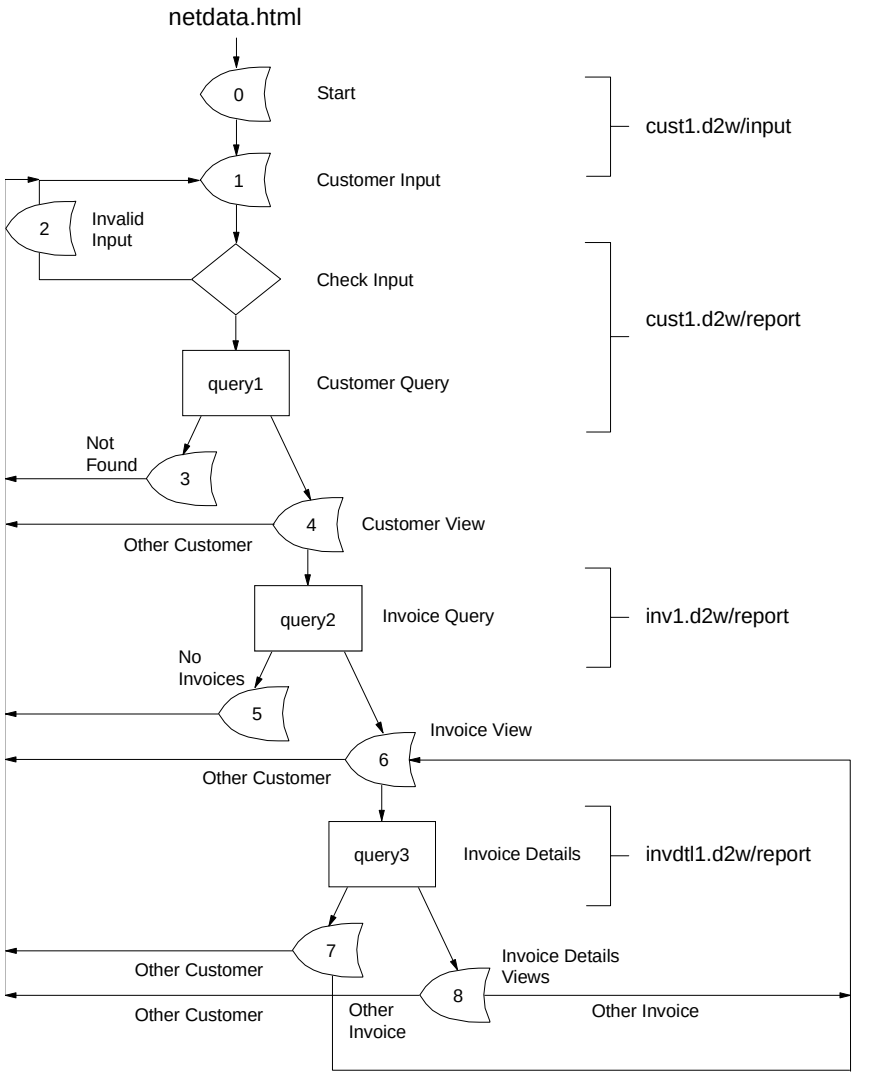
This example is taken from the labs developed by Jon Rush, for the Advanced Internet Application Development class available to AS/400 Business Partners through AS/400 Partners In Development at IBM, Rochester, MN.

It is a simple but complete Customer Invoice Inquiry application, which also includes some logic control and arithmetic calculations. The main goal of this example is to illustrate that Net.Data is no longer just a SQL extension for the Web.

The test environment consisted of:

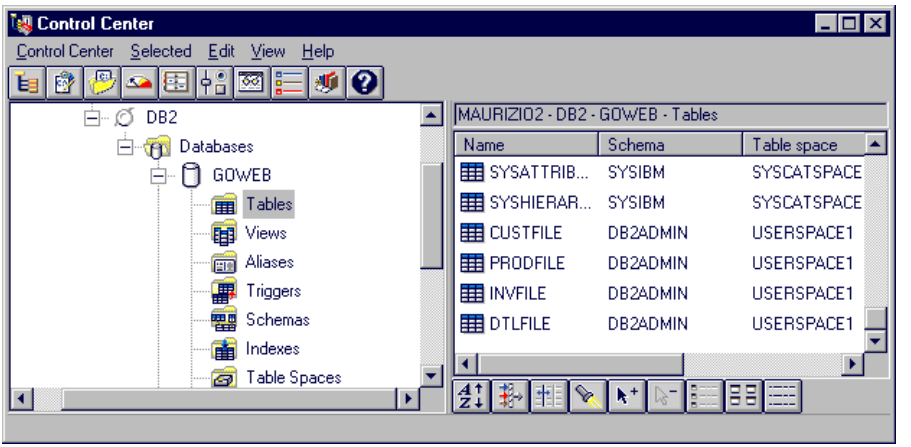
- Server
  - Windows NT Workstation 4.0
  - DB2 Universal Database version 5
  - Domino Go Webserver 4.6
- Client
  - Windows 95
  - Netscape Navigator 3.0

Application flowchart

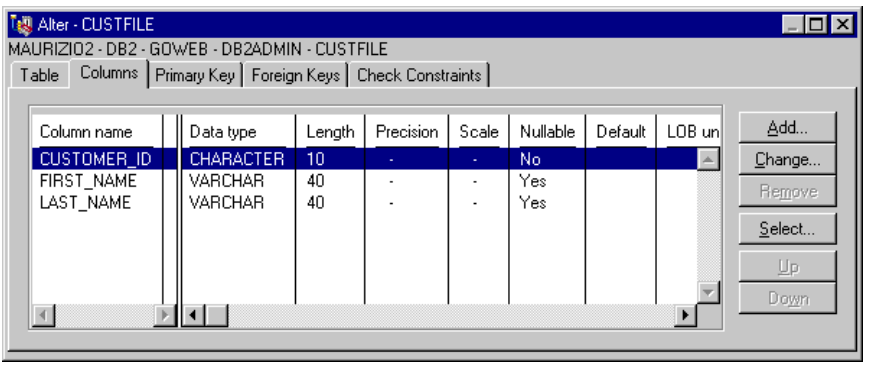


Database

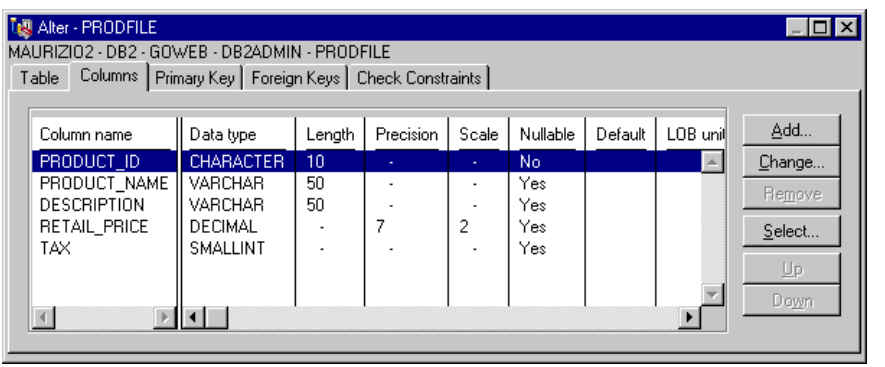
The following DB2 Control Center screens describe the content of the tables of the Database GOWEB used for this example.



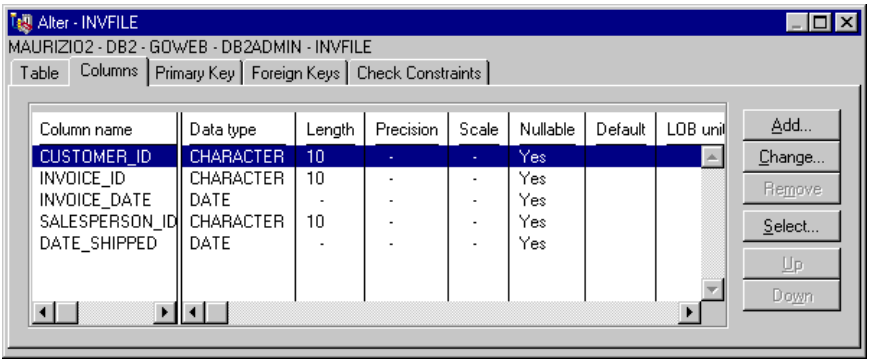
Customers Table



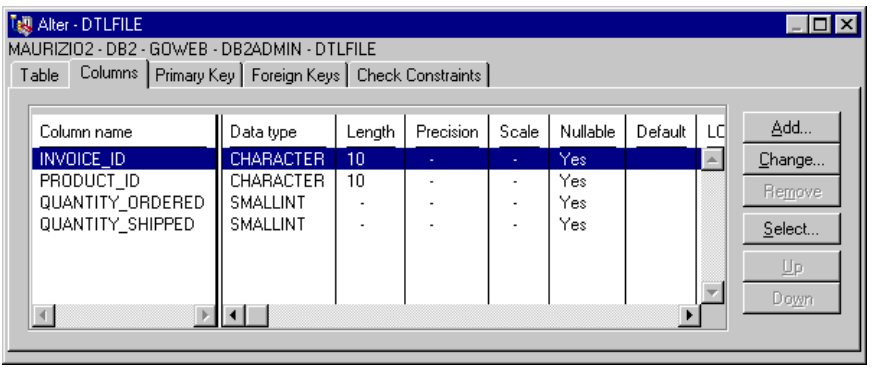
Products Table



Invoices Table



Invoice Details Table



Application code and screens

This section will give a high-level overview of the application and explain its flow and then the Net.Data macros that perform the task are shown.

The code that was used in this example consists of:

- An initial html page
- A Net.Data macro with two entry points for customer inquiries
- Two Net.Data macros with a single entry point for invoice and invoice details inquiries

This Customer Invoice query application begins or is invoked via a hypertext link similar to the following:

```
http://goweb1.lotus.com/cgi-bin/db2www/cust1.d2w/input
```

The Web user is prompted to enter a customer number. When the submit query button is pressed on the HTML form, Net.Data and our macro are invoked by the HTTP server to:

1. Verify that the user entered a value for customer number, or
2. Perform the actual SQL function to query the customer table.

Error messages via HTML pages are returned if the user enters no customer number or if there is no customer number in the customer table that matches what the user has inputted.

If there is a customer in the customer table, the Net.Data application will build an HTML page with the results and two hypertext links.

The first link allows the Web user to query for another customer number. The second link will invoke another Net.Data macro which will present this current customer's invoices from the invoice table. The HTML page prepared by the invoice macro will show, in tabular form, all invoices belonging to this customer. The actual invoice numbers will be hypertext links to allow drill down to get specific invoice detail. A third macro is called to get invoice detail data from the third table. It presents the items in the specific invoice as well as using Net.Data built-in functions to do some arithmetic operations to find the total dollar amount of this invoice.

The Net.Data macros are listed next that accomplish this task. Key things to look for in these macros are the HTML sections designated by %HTML(xxxxx) where xxxxx is the HTML section name and the @xxxx statements which are calls to the processing pieces of the macro. %FUNCTION statements define the functions that are invoked with the @xxxx syntax.

Caution Owing to the length of the instruction, some HTML statements are spread over more than one line. If you want to run this example, please verify the HTML syntax of the statements longer than one line.

netdata.html

```
<HTML>

<head>

<TITLE>Net.Data Example</TITLE>

</head>

<body>

<CENTER>

<table>

<tr><td><img align=left src="/Admin/lgmast.gif" height=46
width=240></td>
```

```
    <td><img align=right src="/images/netdata.gif" height=46
   width=240></td></table>

<p><H1>I.T.S.O. GOWEB1</H1> <P>

<A href="http://goweb1.lotus.com/cgi-bin/db2www/cust1.d2w/input">
        Net.Data Example</A>

</CENTER> <p>

</body>

</HTML>
```

Page 0 - Application Start

First Net.Data Macro - CUST1.D2W

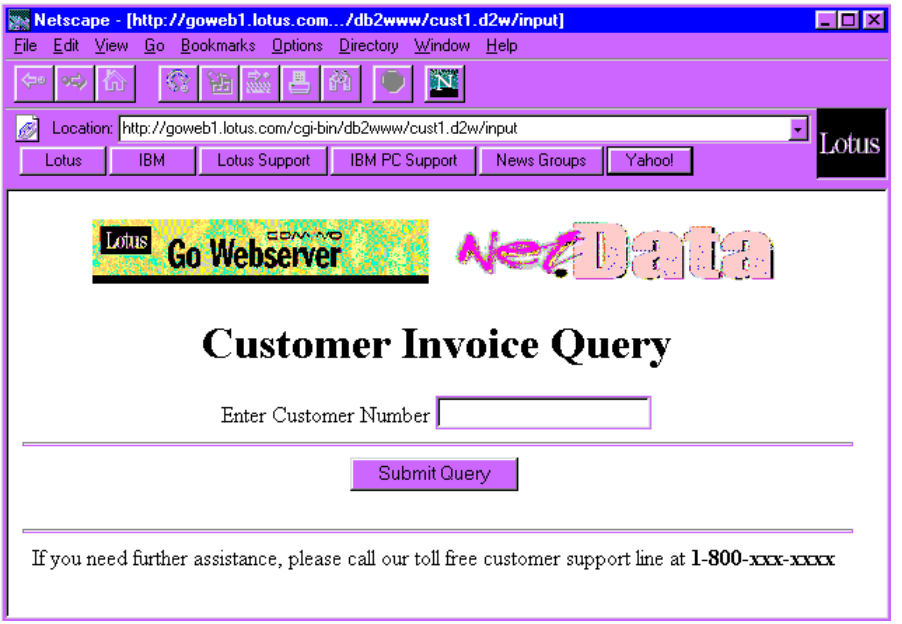
```
%DEFINE {
  DATABASE="GOWEB"
  TABLE="CUSTFILE"
  LOGIN="Db2admin"
  PASSWORD="xxxx"
  result=" "
  fnam=""
```

```
  lnam=""
%}
%{***************** SQL FUNCTION ************%}
%FUNCTION(DTW_SQL) query1() {
   select * from $(TABLE) where CUSTOMER_ID='$(cscus)'
      %REPORT{
        <center>
        <table cellspacing=2 cellpadding=5 border=1>
            <tr><th>Customer #</th>
                   <th>First Name</th>
                   <th>Last Name</th></tr>
            %ROW{
                <tr><td>$(V1)</td><td>$(V2)</td><td>$(V3)</td></tr>
                  @DTW_ASSIGN(fnam, V2)
                   @DTW_ASSIGN(lnam, V3)
            %}
        </table>
        </center>
      %}
    %MESSAGE {
      100: "<center><b><i>No Customer found with customer number:
          $(cscus)</i></b>" : continue
       -100: "<center><b><i>No Customer found with customer number:
            $(cscus)</i></b>" : continue
      %}
%}
%{************** HTML INPUT **********%}
%HTML_INPUT{
  <HTML>
  <BODY >
  <CENTER>
  <table>
```

```
  <tr><td><img align=left src="/Admin/lgmast.gif" height=46
              width=240></td>

      <td><img align=right src="/images/netdata.gif" height=46
              width=240></td></table>

  <H1>Customer Invoice Query</h1>


  <FORM Method=POST ACTION="/cgi-bin/db2www/cust1.d2w/report">

  Enter Customer Number

  <INPUT TYPE=text NAME=cscus><hr>

  <INPUT TYPE=submit>

  </FORM>

  </center>

  <hr>

  <center>

  <font size=3>If you need further assistance, please call our toll
         free customer support line at <B>1-800-xxx-xxxx</b></font>

  </center>

  </body>

  </HTML>

%}

%{******************** HTML REPORT ********************%}

%HTML (report){

  <HTML>

  <BODY >

  <TITLE>Query Results</TITLE>

  <CENTER>

  <table>

  <tr><td><img align=left src="/Admin/lgmast.gif" height=46
              width=240></td>

      <td><img align=right src="/images/netdata.gif" height=46
              width=240></td></table>

  <P>

  Here is the result of your request

  </center><P>
```

```
%IF ("$(cscus)" == "")

  <P><center><B>No customer # entered</B><P>

  <Font size=4><a href="/cgi-bin/db2www/cust1.d2w/input">View Another
                        Customer</a></font>

 %ELIF  ("$(cscus)" == "0")

  <P><center><B>Customer # 0 not allowed</B> <P>

  <Font size=4><a href="/cgi-bin/db2www/cust1.d2w/input">View Another
                      Customer</a></font>

%ELSE
 @query1()

 <HR><center><table cellspacing=20 border=1>

 <tr><td>

     <a href="/cgi-bin/db2www/inv1.d2w/report?incus=$(cscus)
        &infnam=$(fnam) &inlnam=$(lnam)">Click here to see
        outstanding invoices</a></td>

 <td><a href="/cgi-bin/db2www/cust1.d2w/input">View Another
             Customer</a></td></tr>

 </table>

%ENDIF

 </center>

 </body>

 </HTML>

%}
```

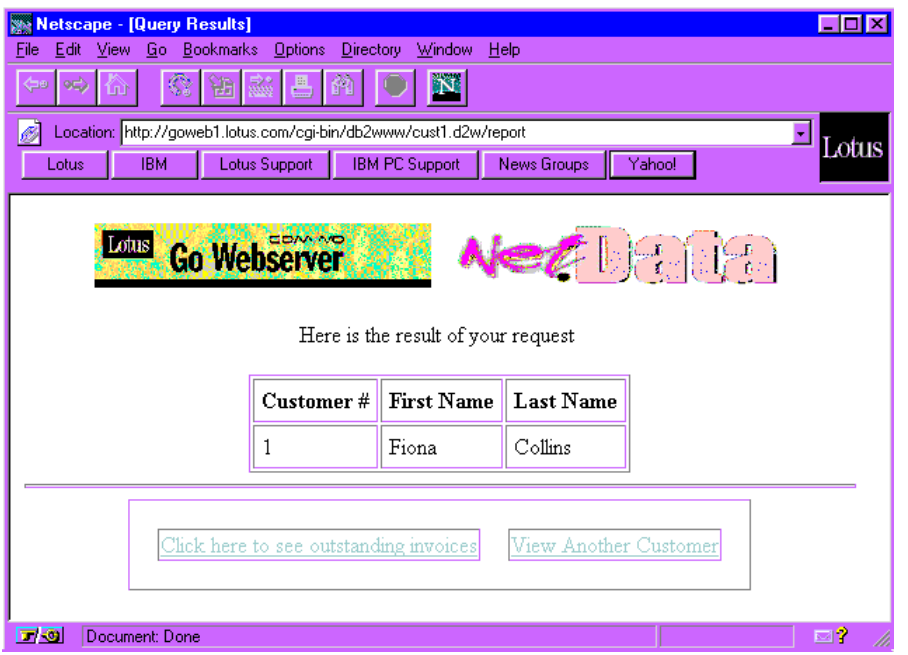
Page 1 - Customer Input (output of CUST1.D2W/INPUT)



Page 2 - Invalid Input (output of CUST1.D2W/REPORT before query)

Page 3 - Customer # not found (output of CUST1.D2W/REPORT after query)

Page 4 - Customer View (output of CUST1.D2W/REPORT)



Second Net.Data Macro: INV1.D2W

```
%DEFINE {
  DATABASE="GOWEB"
  TABLE="INVFILE"
  LOGIN="Db2admin"
  PASSWORD="xxxx"
  result="0"
%}
%{******************** SQL FUNCTION ******************%}
%FUNCTION(DTW_SQL) query2() {
select * from $(TABLE) where CUSTOMER_ID = '$(incus)' ORDER
  BY INVOICE_ID
      %REPORT{
            <center>
            <table cellspacing=2 cellpadding=5 border=2>
            <tr><th>Invoice</th>
               <th>Invoice Date</th>
               <th>Sales Person</th>
               <th>Date Shipped</th>
</tr>
            %ROW{
             <tr> <td><ahref="/cgi-bin/db2www/invdtl.d2w/report?
                      invnum=$(V2)&infnam=$(infnam)&inlnam=$(inlnam)
                      &incusnum=$(incus)">$(V2)</a></td>
             <td>$(V3)</td><td>$(V4)</td><td>$(V5)</td></tr>
            %}
            </table>
            </center>
      %}
      %MESSAGE {
            100: "<center><b><i>No Invoices found for customer
            number: $(incus)</i></b>" : continue
```

```
            -100: "<center><b><i>No Invoices found for customer
             number: $(incus)</i></b>" : continue

      %}

 %}

%{************   HTML REPORT    ***********%}

%HTML (report){

  <HTML>

  <BODY>

  <TITLE>Query Results</TITLE>

  <center><table>

  <tr><td><img align=left src="/Admin/lgmast.gif" height=46
               width=240></td>

      <td><img align=right src="/images/netdata.gif" height=46
               width=240></td></table>

  </center>

  <P>

  <center><H4>Here are the outstanding invoices for <b> $(infnam)
              $(inlnam) / Customer #: $(incus) </H4><br>

        Click on the Invoice number to see details</b>

</center>

  <HR>

  <P>

  @query2()

  <HR>

  <center><a href="/cgi-bin/db2www/cust1.d2w/input">View
            Another Customer</a></center>


  </body>

  </HTML>

%}
```

Page 5 - No invoices found

Page 6 - Invoices View

Third Net.Data Macro - INVDTL.D2W

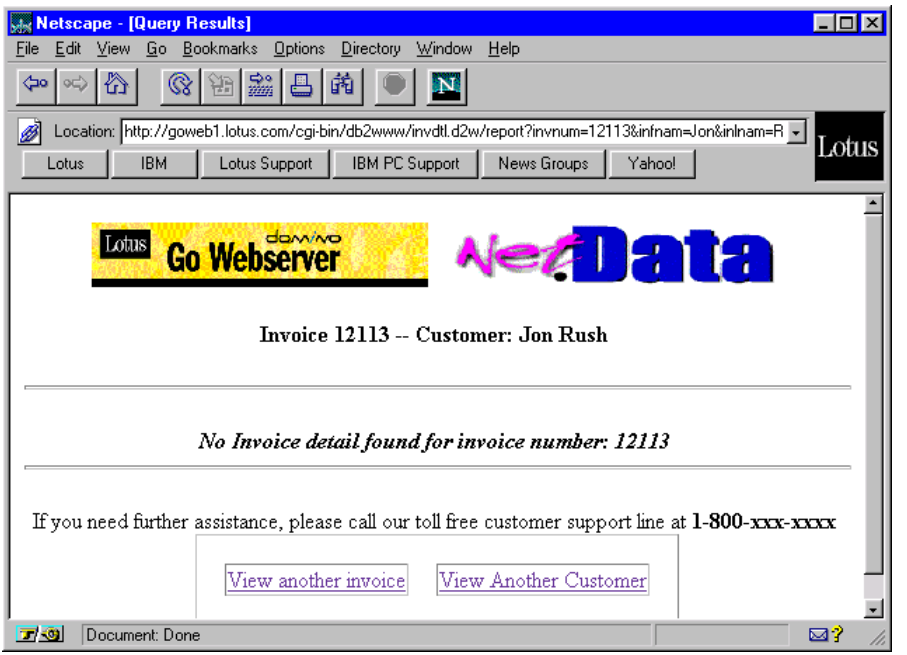
```
%DEFINE {
  DATABASE="GOWEB"
  TABLE1="DTLFILE"
  TABLE2="PRODFILE"
  LOGIN="Db2admin"
  PASSWORD="xxxx"
  result="0"
  unittotal="0"
  taxrow="0"
  taxtotal="0"
  total="0"
  hundred="100"
```

```
  SHOWSQL="NO"
  zeros="0.00"
%}
%{******************** SQL FUNCTION ********************%}
%FUNCTION(DTW_SQL) query3() {
   SELECT * FROM $(TABLE1) A , $(TABLE2) B WHERE A.INVOICE_ID =
'$(invnum)' AND
    A.PRODUCT_ID = B.PRODUCT_ID ORDER BY A.PRODUCT_ID
      %REPORT{
            <center>
            <table cellspacing=1 cellpadding=1 border=1>
            <tr><th>IBM #</th><th>Lotus #</th><th>Description</th>
                  <th>Unit Price</th><th>Q.ty</th><th>Tax %</th>
                  <th>Unit Total</th></tr>
            %ROW{
                  @DTW_MULTIPLY(V8,V4,unittotal)
                <tr><td>$(V5)</td><td>$(V6)</td><td>$(V7)</td>
                    <td>$(V8)</td><td>$(V4)</td><td>$(V9)</td>
                  <td align=left>@DTW_rMULTIPLY(V8,V4)</td></tr>
                  @DTW_ADD(result,unittotal, result)
                  @DTW_MULTIPLY(unittotal,V9,taxrow)
                  @DTW_DIVIDE(taxrow,hundred,taxrow)
                  @DTW_ADD(taxrow,taxtotal,taxtotal)
            %}
                  @DTW_ADD(taxtotal,result,total)
              <tr><td></td><td></td><td><b>Total</b></td><td></td>
                  <td></td><td></td><td><b>$(result)</b></td></tr>

              <tr><td></td><td></td><td><b>Tax Total</b></td><td></td>
                  <td></td><td></td><td><b>$(taxtotal)</b></td></tr>
             <tr><td></td><td></td><td><b>Invoice Total</b></td>
              <td></td><td></td><td></td><td><b>$(total)</b></td></tr>
```

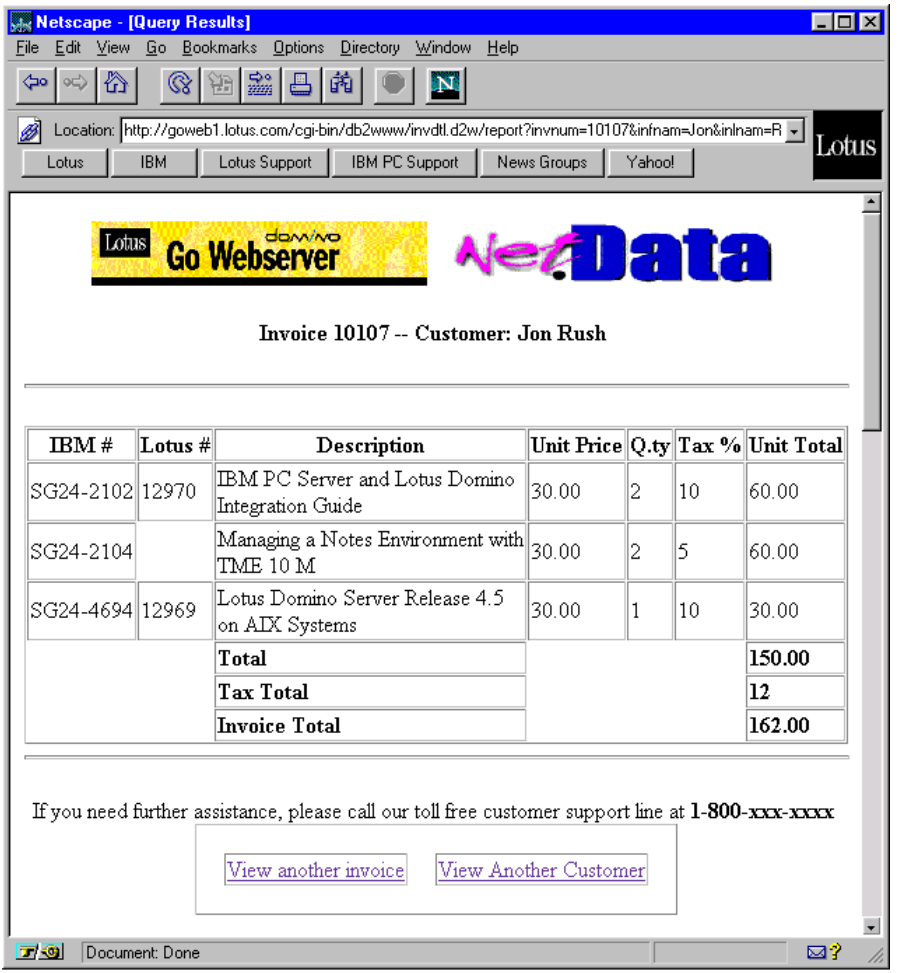
```
            </table>

            </center>

      %}

      %MESSAGE {

       100: "<center><b><i>No Invoice detail found for invoice number:
              $(invnum)</i></b>" : continue

      -100: "<center><b><i>No Invoice detail found for invoice number:
              $(invnum)</i></b>" : continue

       %}

%}

%{******************** HTML REPORT ********************%}

%HTML (report){

  <HTML>

  <TITLE>Query Results</TITLE>

  <BODY>
  <center><table>

  <tr><td><img align=left src="/Admin/lgmast.gif" height=46
               width=240></td>

      <td><img align=right src="/images/netdata.gif" height=46
       width=240></td></table>

  <center><H4> Invoice $(invnum) -- Customer: $(infnam) $(inlnam)
    </h4></center>

  <HR>

  <P>

  @query3()

  <HR>

  <br><center>

 <font size=3>If you need further assistance, please call our toll
          free customer support line at <B>1-800-xxx-xxxx</b></font>

  <center><table cellspacing=20 border=1>

  <tr><td><a href="/cgi-bin/db2www/inv1.d2w/report?incus=$(incusnum)
     &infnam=$(infnam)&inlnam=$(inlnam)">View another invoice</a></td>

      <td><a href="/cgi-bin/db2www/cust1.d2w/input">View Another
     Customer</a></td></tr>

  </table>
```

```
  </center>
  </body>
  </HTML>
%}
```

Page 7 - No Details Found



Page 8 - Invoice Details



Appendix D
Domino Go Webserver HTTPD.CNF Configuration File

The following is an excerpt from the Domino Go Webserver configuration file showing the various performance directives.

```
# ====================================================================
# ====================================================================
#        Default configuration file for httpd, running as a normal
#        HTTP server.
#        TABLE OF CONTENTS
#=====================================================================
# ====================================================================
#        - Basic directives
#        - Logging and Reporting directives
#          * log purge/archive directives
#          * access log filter directives
#          * example report templates
#        - Method directives
#        - Directories and Welcome page directives
#          * Directory browsing directives
#        - Error Message Customization
#        - User directory directives
#        - GWAPI directives:
#        - User authentication and document protection
#        - Service directives
#          * HTImage directives
#          * HTCounter directives
#        - Mapping rules
#        - Performance directives.
#        - Timeout directives
```

```
#        - Security directives.
#        - Proxy directives
#        - Proxy caching directives
#        - File caching directives
#        - SNMP directives
#        - Icon directives
#        - Request Processing directives
#------------------------------------------------------------------------
#------------------------------------------------------------------------
#        DNS-Lookup directive:
#------------------------------------------------------------------------
#------------------------------------------------------------------------
#        Instruct the server to look up hostnames of clients by
#        setting DNS-Lookup to "on".
#        NOTE: Turning DNS-Lookup "on" decreases server performance.
#        Default:  off
#        Syntax:   DNS-Lookup <on | off>
DNS-Lookup     off
#------------------------------------------------------------------------
#------------------------------------------------------------------------
#        imbeds directive:
#------------------------------------------------------------------------
#------------------------------------------------------------------------
#        Controls Server Side Include processing for output that has
#        Content-type: text/x-ssi-html.
#        Default:  off SSIOnly
#        Syntax:   imbeds <on/off/files/cgi/noexec> <html/SSIOnly>
#        parm1: on         process files, CGIs and SSI
#                          exec CGI
#                          off        never use SSI
#                          files      process files and SSI #exec CGI
```

```
#                       cgi        process CGIs  and SSI #exec CGI
#                       noexec     process files, CGIs but not SSI
#                       exec CGI
#       parm2: html       also process Content-type: text/html
#       SSIOnly    process Content-type: text/x-ssi-html only
#NOTE: If more than one imbeds directive is specified, the last one is
#      used.
imbeds on SSIOnly
=====================================================================
=====================================================================
#        Logging and Reporting directives
=====================================================================
=====================================================================
#        If you want logging, specify locations for your logs:
#           AccessLog      - used for logging local document requests
#           AgentLog       - used for logging browser requests
#           RefererLog     - used for logging requests which are
#                            refered
#           ErrorLog       - used for logging any errors
#           CgiErrorLog    - used for logging any CGI errors
#           ProxyAccessLog - used for logging proxy requests
#           CacheAccessLog - used for logging hits on proxy cache
#                            (only valid if server running as proxy)
#NOTE: To enable logging of requests to the proxy cache, the
#      following must be defined:
#    Caching MUST be turned ON (default is OFF)
#    CacheRoot MUST be defined (by default, no CacheRoot is defined)
#    CacheAccessLog MUST be defined
#    Defaults:  AccessLog     d:\www\logs\httpd-log
#                   AgentLog      d:\www\logs\agent-log
#                   RefererLog    d:\www\logs\referer-log
#                   ErrorLog      d:\www\logs\httpd-errors
#                   CgiErrorLog   d:\www\logs\cgi-error
```

```
#        Syntax:    <directive> <fullpath-filename>
# Example:
# ProxyAccessLog   d:\www\logs\httpd-proxy
# CacheAccessLog   d:\www\logs\httpd-cache
AccessLog C:\WWW\Logs\httpd-log
AgentLog C:\WWW\Logs\agent-log
RefererLog C:\WWW\Logs\referer-log
ErrorLog C:\WWW\Logs\httpd-error
CgiErrorLog C:\WWW\Logs\cgi-error
#        LogFormat directive:
#        Default:  Common
#        Syntax:   LogFormat  <Old | Common>
LogFormat     Common
#        LogTime directive:
#        Default:  LocalTime
#        Syntax:   LogTime  <GMT | LocalTime>
LogTime       LocalTime
#        LogToGUI directive:
#        Display log entries in the HTTP server window
#        on the workstation. To optimize server performance,
#        this option is set to "off".
#        Default:  off
#        Syntax:   LogToGUI <on | off>
LogToGUI on
#        NoLog directive:
#        Suppress access log entries for host matching a given IP
#        address or hostname. Wild cards "*" may be used. This
#        directive may be used
#        multiple times within the configuration file.
#        Default:  <none>
#        Syntax:   NoLog  <hostnames and IP addresses>
#        NOTE: DNS-Lookup may need to be turned ON.
```

```
# Example:

# NoLog    128.141.*.*

# NoLog    *.location.company.com

# Nolog    *.*.*.com

#=============================================================================

#   *** log purge/archive directives ***

#=============================================================================

#        AccessLogArchive and ErrorLogArchive directive:

#        Enables the purge options (purge) or the user exit option
#        (userexit) or does not do either (none). When selected, the
#        purge action or

#        userexit action will take place at midnight, immediately
#        after the

#        previous day's logs have been closed and the new day's logs
#        have been

#        opened. If the userexit option is specified, the name and
#        location

#        of the user exit that is called must be specified following
#        the userexit option parameter.

#        Default:  AccessLogArchive none

#                  ErrorLogArchive  none

#        Syntax:   AccessLogArchive <none | purge | userexit
#                  user_exit_spec>

#                  ErrorLogArchive  <none | purge | userexit
#                  user_exit_spec>

# Example:

# AccessLogArchive purge

# ErrorLogArchive  purge

# AccessLogArchive userexit d:\usercode\movelogs.exe -d -o -g

# ErrorLogArchive  userexit d:\usercode\movelogs.exe -d -o -g

AccessLogArchive none

ErrorLogArchive  none

#        AccessLogExpire and ErrorLogExpire directive:

#        Sets the age limit, in DAYS, for access log files. Any
#        access/error
```

```
#        log files older than the number of days specified will be
#        erased. If

#        set to zero then no expiration date exists. The file creation
#        date as

#        reported by the operating system is used to determine the
#        date - the

#        suffix of the filename (such as httpd-log.Mar2297) is not
#        used to

#        determine file age.

#        Default:  AccessLogExpire 0

#                  ErrorLogExpire  0

#        Syntax:   AccessLogExpire <num>

#                  ErrorLogExpire  <num>

# Example:

# AccessLogExpire 30

# ErrorLogExpire  10

AccessLogExpire 0

ErrorLogExpire  0

#        AccessLogSizeLimit and ErrorLogSizeLimit directive:

#        Sets the sum total size limit, in megabytes, for the
#        access/error logs.

#        If set to zero, then no maximum size is enforced. This
#        directive takes

#        effect after AccessLogExpire/ErrorLogExpire has been applied.
#        If the sum total size of the corresponding log files is
#        larger than the limit

#        assigned, files will be deleted starting with the oldest file
#        until the

#        total size is within the limit.

#        Default:  AccessLogSizeLimit 0

#                  ErrorLogSizeLimit  0

#        Syntax:   AccessLogSizeLimit <num>

#                  ErrorLogSizeLimit  <num>

# Example:

# AccessLogSizeLimit 20
```

```
# ErrorLogSizeLimit  10

AccessLogSizeLimit 0

ErrorLogSizeLimit  0

#=====================================================================

#   *** access log filter directives ***

#=====================================================================

#        AccessLogExcludeURL, AccessLogExcludeMethod,

#        AccessLogExcludeReturnCode and AccessLogExcludeMimeType
#        directive:

#        Access log entries may be filtered to exclude requests by:

#           * requests matching a given URL template

#           * requests of a given method

#           * requests with a given return code range (200s, 300s,
#             400s, 500s)

#           * requests for files of a given mime type

#        Default:  <none>

#        Syntax:   AccessLogExcludeURL    <URL template>

#                  AccessLogExcludeMethod <GET | PUT | POST | DELETE>

#                  AccessLogExcludeReturnCode <200 | 300 | 400 | 500>

#                  AccessLogExcludeMimeType <text/html | text/plain |

#                                            text/other | image/gif  |

#                                            image/jpeg |
#                                            image/(other) |

#                                            application/* |

#                                            audio/* | video/* |

#                                            (other)/(other)>

# Example:

# AccessLogExcludeURL  *.gif

# AccessLogExcludeURL  /Freebies/*

# AccessLogExcludeMethod PUT

# AccessLogExcludeMethod POST

# AccessLogExcludeReturnCode 300
```

```
# AccessLogExcludeReturnCode 400
# AccessLogExcludeMimeType text/html
# AccessLogExcludeMimeType text/plain
========================================================================
========================================================================
#  *** Directory browsing directives ***
========================================================================
========================================================================
#        These directives control directory listings as follows:
#          * Enable/disable or selective directory browsing
#          * Configure/disable readme feature for directory browsing
#          * Control the appearance of the directory listing
#          * Define the maximum width of the description text
#          * Define the maximum & minimum width of the filename field
#        Default:  DirAccess           off
#        Syntax:   DirAccess <on | off | selective>
#        Default:  DirReadme           top
#        Syntax:   DirReadme <top | bottom | off>
#        Default:  DirShowIcons        on
#                  DirShowDate         on
#                  DirShowSize         on
#                  DirShowDescription  on
#                  DirShowBrackets     on
#                  DirShowCase         on
#                  DirShowHidden       on
#                  DirShowBytes        off
#        Syntax:   <directive> <on | off>
#        Default:  DirShowMaxDescrLength 25
#                  DirShowMaxLength      25
#                  DirShowMinLength      15
#        Syntax:   <directive> <num>
```

```
DirAccess               On
DirReadme               top
DirShowIcons            on
DirShowDate             on
DirShowSize             on
DirShowDescription      on
DirShowBrackets         on
DirShowCase             off
DirShowHidden           on
DirShowBytes            off
DirShowMaxDescrLength   25
DirShowMaxLength        25
DirShowMinLength        15
#       UseMetaFiles directive:
#       Specify whether the meta files are used by the server.
#       Set this directive to "off" for better server performance;
#       set to "on" if you're using meta-information files.
#       Default:  off
#       Syntax:   UseMetaFiles <on | off>
UseMetaFiles off
#       MetaDir and MetaSuffix directive:
#       Specify the directory where meta-information should be stored
#       and the suffix for the file in which meta-information is to
#       be
#       stored. Meta-information is stored in a file with the same
#       name
#       as the actual document, but with the suffix specified by the
#       MetaSuffix directive.
#       Default:  .web
#       Syntax:   MetaDir  <directory name>
#       Default:  .meta
#       Syntax:   MetaSuffix  <.suffix>
```

```
MetaDir      .web

MetaSuffix   .meta

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

#        Performance directives.

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

#        MaxActiveThreads directive:

#        Defines the maximum number of threads in system thread pool.

#        Default:  40

#        Syntax:   MaxActiveThreads  <num>

MaxActiveThreads  40

#        MaxPersistRequest directive:

#        Maximum number of request to receive on a persistent
#        connection.

#        Default:  5

#        Syntax:   MaxPersistRequest <num>

MaxPersistRequest 5

#        ServerPriority directive:

#        Default:  1

#        Syntax:   ServerPriority  <0 | 1 | 2>

#        Note:    This is the priority on your system you want your
#                 server to run.

#                 0 -  background process (no priority)

#                 1 -  maximum priority as a background process

#                 2 -  maximum priority as a foreground process.

ServerPriority 0

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

#        Timeout directives

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

#        Use these directives to:

#           * limit the time to wait for the client to send a request
```

```
#             after connecting to the server before cancelling the
#             connection.
#           * limit the time to allow for sending output to the
#             client.
#           * limit the time to allow for server scripts to finish.
#             (If the program does not finish within allotted time,
#             the server
#              will send a TERM signal and then a KILL signal 5
#              seconds later
#              to stop the program.)
#           * limit the time to wait for the client to send a request
#             after establishing a persistent connection to the server
#             before cancelling the connection.
#        Default:  InputTimeout    2 minutes
#        Default:  OutputTimeout  20 minutes
#        Default:  ScriptTimeout   5 minutes
#        Default:  PersistTimeout  1 minute
#        Syntax:   <directive> <time-spec>
InputTimeout    2 minutes
OutputTimeout  20 minutes
ScriptTimeout   5 minutes
PersistTimeout  1 minute
 ============================================================================
============================================================================
#        File caching directives
============================================================================
============================================================================
#        LiveLocalCache directive:
#        Specify whether or not the local cache is updated when a
#        cached
#        file is modified. Specify On if you want users requesting a
#        cached
#        file to get the file with the latest updates. Off is the high
#        performance setting.
```

```
#        Default:  off
#        Syntax:   LiveLocalCache <on | off>
LiveLocalCache off
#        CacheLocalMaxFiles and CacheLocalMaxBytes directive:
#        Use these directives to:
#           * limit the number of files which can be cached in memory
#           * limit the amount of memory used to cache files in memory
#        These are useful when wildcards are used in CacheLocalFile
#        directives. Use a value of 0 to indicate no maximum.
#        Default:  200
#        Syntax:   CacheLocalMaxFiles <num>
#        Default:  2 M
#        Syntax:   CacheLocalMaxBytes <num> <K | M>
CacheLocalMaxFiles 200
CacheLocalMaxBytes 2 M
#        CacheLocalFile directive:
#        Path and name of files that are to be loaded into memory each
#        time the
#        server is started. This directive may occur multiple times
#        within the
#        configuration file. The name must be fully qualified and may
#        NOT contain
#        any wildcard characters.
#        The URL is optional. If you tell us the URL corresponding to
#        this file,
#        response time will improve significantly.
#        Default:  CacheLocalFile   Frntpage.html   /Frntpage.html
#                  CacheLocalFile   lgmast.gif      /Admin/lgmast.gif
#                  CacheLocalFile   lgsplash.gif
/Admin/lgsplash.gif
#        Syntax:   CacheLocalFile  <file path> <URL>
# Example:
# CacheLocalFile   d:\example\path\index.html /index.html
```

```
CacheLocalFile  C:\WWW\HTML\Frntpage.html
CacheLocalFile  C:\WWW\Admin\lgmast.gif
CacheLocalFile  C:\WWW\Admin\lgsplash.gif
```

Appendix E
Domino HTTPD.CNF File

Domino uses three different files to determine its configuration:

1. The Public Address Book
2. The domcfg.nsf database
3. httpd.cnf

This appendix focuses on the relationship between the Domino Public Address Book and the httpd.cnf file directives.

As a general rule, if parameter values are located in both the server document and the httpd.cnf file you should use the server document to set its value and the directive should not be coded in the httpd.cnf file. One exception to this rule was discovered during testing which is the "welcome" directive. Adhering to this general rule ensures that Domino is using the values coded in the server document. If it is set in both places, depending on which parameter or directive it is, Domino may use the value in the server document or the one in the httpd.cnf file. The reason is that some parameters can be set more than once and only the final value is used, while for others the first value found is used and others are ignored. There is no listing of which parameters fall into which category, so it's best to avoid conflicts where possible.

Please refer to the *Domino Go Webserver Webmaster's Guide* for a detailed description of each directive listed.

HOMDIR
Does not appear in the server document. Modify in the httpd.cnf file.

USERDIR
Does not appear in the server document. Modify in the httpd.cnf file.

Server Side Includes
Server side includes work with Domino in the same way as with Domino Go Webserver. The ADDTYPE directives that are needed to enable this function are already provided in the htpd.cnf file that is shipped with Domino.

PORT

This parameter is found in both the server document and the httpd.cnf file. This value should be coded in the server document. If coded in the httpd.cnf file, this value will override the one in the server document.

WELCOME Page

The Welcome page directive in the httpd.cnf file has been renamed the "default home page" in the Domino server document. This directive is the one exception to the rule regarding never coding a directive in both the server document and the httpd.cnf file. Domino first looks through all the Welcome directives in the httpd.cnf file and only when it fails to find a file that matches one of the Welcome directives will it use the "default home page" parameter value in the server document.

The benefit of coding the Welcome directive in the httpd.cnf file is that multiple Welcome directives can be listed and Domino will search for a file in the order that they are listed. Only one default home page can be coded in the server document. However, if you want the default home page to be a Notes database open, or a Notes document, you must specify this in the default home page field in the server document.

Server Logs

There are five logs that can be specified in the server document. These logs can also be specified as directive statements in the httpd.cnf file. In addition, the server document has another field, Enable Logging, which can be enabled or disabled. The five logs have similar but not identical names.

Domino Name	*Domino Go 'httpd.cnf" Directive Name*
Access Log	AccessLog
Agent Log	AgentLog
Refer Log	ReferrerLog
Error Log	ErrorLog
CGI Error Log	CgiErrorLog

Several combinations of specifying the logs in both the server document and the httpd.cnf file were tested while setting the "Enable Logging" field in the server document to both enable and disable.

Enable Logging Field	*Server Document*	*HTTPD.CNF File*	*Results*
Enabled	Yes	Yes	Uses server document
Disabled	Yes	Yes	Uses httpd.cnf file
Enabled	No	Yes	Uses httpd.cnf file
Disabled	Yes	No	No logging done

As a rule it's best to code the log file names in the server document. The only benefit to coding the directives in the httpd.cnf file is that you can specify a different directory for each log file. Only one directory can be specified for all five log files in the server document.

NOLOG

The NOLOG directive provides the option not to log requests from specific hosts or domains. It can be set in the server document and the httpd.cnf file. If it's coded in both the server document and httpd.cnf, it uses the value in the server document. If it's only set in the httpd.cnf file, it uses that value.

DNS Lookup

This function only works if coded in the server document.

LogTime

This function only works if coded in the server document.

Appendix F
Domino Go Webserver and Domino Product Comparisons

The following table provides some guidance on the differing features of the Domino and Domino Go Webserver products:

Feature	*Domino Go*	*Domino*
Search Engine	Net.Question	Verity
APIs	GWAPI	Notes API for C, C++, LotusScript classes, Java classes
NSAPI	No	No
ISAPI	No	No
CGI	Yes	Yes
Security		
SSL3	Yes	Yes
SSL Tuneling	Yes	?
X.509v3 Certificate Creation	Yes (Cert utility)	Yes
X509v3 Certificate Support	Yes	Yes
Certificate Authority (CA)	Yes (private)	Yes (Private)
PICS - Rating Service	Yes	No
Field Level Access Control	No	Yes
File Level access control	Yes	Yes
Proxy		
Proxy server for browser	Yes	?
Proxy server caching	Yes	?
Deny List	?	Yes
Application Development		
Integrated Development Enviroment	No	Yes
Lotus BeanMachine	Yes	Yes

continued

Feature	*Domino Go*	*Domino*
NetObjects Fusion	Yes	No
Content Store	File System	NSF Rich Object Store
Server Side Includes	Yes	Yes
Internet Cookies	Yes	Yes
Java Support		
JDK 1.1	Yes	Yes
Java Servlet support	Yes	Yes
Java Applications	Yes	Yes
Java Applets	Yes	No
Java Agents	N/A	Yes
Protocols		
HTTP 1.1	Yes	Yes
SMTP	No	Yes
MIME	No	Yes
POP3	No	Yes
IMAP4	No	Yes
LDAP	No	Yes
NNTP	No	Yes
SNMP	Yes	Yes
SSL3	Yes	Yes
PICS	Yes	No
Administration & Management		
SNMP	Yes	Yes
Browser Based Administration	Yes	Yes
NT Performance Monitor	Yes	Yes
NT Event Viewer	Yes	Yes
Configurable Log Files	Yes	Yes
Common Log Format	Yes	?
Web Usage Mining	Yes	? (user activity log)
Report of Stats	Yes	Yes (Stats databse)

continued

Feature	*Domino Go*	*Domino*
Configuration	Yes httpd.cnf file	Yes public name & addres book httpd.cnf domcfg.nsf database
Platforms		
NT Intel	Yes	Yes
NT Allpha	No	Yes
Win95	Yes	Yes
OS/2 Warp	Yes	Yes
AIX	Yes	Yes
Sun Solaris SPARC	Yes	Yes
HP-UX	Yes	Yes
OS400	No	Yes
OS390	Yes	Yes
Other		
Multi-homing (virtual servers)	Yes	Yes

Special Notices

This publication is intended to help administrators plan for and deploy Domino Go Webserver.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM products, programs, or services may be used. Any functionally equivalent program that does not infringe any IBM intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendors and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific

information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other trademarks are trademarks of their respective companies.

The following are trademarks of Lotus Development Corporation in the United States and/or other countries.

DataLens®
InterNotes
InterNotes Web Publisher
Lotus®
Lotus Notes Reporter
Lotus Notes®
Lotus Notes ViP®
Lotus @SQL®
LotusScript®
Notes
Notes HiTest
Notes ViP®
Notes Mail®
NotesPump
NotesSQL
Notes/FX
Phone Notes®
Phone Notes Mobile Mail
SmartIcons®
Video Notes
Word Pro®

Related ITSO Publications

This section lists other Lotus related publications produced by the International Technical Support Organization (ITSO). For information on ordering these ITSO publications see "How To Get ITSO Redbooks."

ITSO Lotus Publications

- *Lotus Notes 4.5: A Developers Handbook*, IBM form number SG24-4876-00, Lotus part number AA0425
- *LotusScript for Visual Basic Programmers*, IBM form number SG24-4856-00, Lotus part number 12498
- *Secrets to Running Lotus Notes: The Decisions No One Tells You How to Make*, IBM form number SG24-4875-00, Lotus part number AA0424
- *Enterprise Integration with Domino.Connect*, IBM form number SG24-2181-00, Lotus part number 12913
- *Deploying Domino in an S/390 Environment*, IBM form number SG24-2182-00, Lotus part number 12957
- *Developing Web Applications Using Lotus Notes Designer for Domino 4.6*, IBM form number SG24-2183-00, Lotus part number 12974
- *The Next Step in Messaging: Case Studies on Lotus cc:Mail to Lotus Domino and Lotus Notes*, IBM form number SG24-5100-00, Lotus part number 12992

Other ITSO Lotus Related Publications

The publications listed in this section may also be of interest:

- *The Domino Defense: Security in Lotus Notes and the Internet*, IBM form number SG24-4848-01, Lotus part number 12967
- *Lotus Solutions for the Enterprise, Volume 1. Lotus Notes: An Enterprise Application Platform*, IBM form number SG24-4837-00, Lotus part number 12968
- *Lotus Solutions for the Enterprise, Volume 3. Using the IBM CICS Gateway for Lotus Notes*, IBM form number SG24-4512-00
- *Lotus Solutions for the Enterprise, Volume 4. Lotus Notes and the MQSeries Enterprise Integrator*, IBM form number SG24-2217-00
- *Lotus Domino Server Release 4.5 on AIX Systems Installation, Customization and Administration*, IBM form number SG24-4694-01, Lotus part number 12969

- *IBM PC Server and Lotus Domino Integration Guide*, IBM form number SG24-2102-01, Lotus part number 12970
- *Using Lotus Notes on the IBM Integrated PC Server for AS/400*, IBM form number SG24-4779-00
- *Mail Integration for Lotus Notes 4.5 on the IBM Integrated PC Server for AS/400*, IBM form number SG24-4977-01
- *Managing a Notes Environment with TME 10 M*, IBM form number SG24-2104-00
- *Image and Workflow Library: Integrating IBM FlowMark with Lotus Notes*, IBM form number SG24-4851-00
- *Lotus Notes Release 4 In a Multiplatform Environment*, IBM form number SG24-4649-00
- *Implementing LAN Server for MVS in a Lotus Notes Environment*, IBM form number SG24-4741-00
- *Using ADSM to Back Up Databases*, IBM form number SG24-4335-02
- *NetFinity V5.0 Database Support*, IBM form number SG24-4808-00
- *An Approach to ODBC: Lotus Approach to DB2*, IBM form number SG24-4685-00

Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. Order a subscription and receive updates 2-4 times a year at significant savings.

CD-ROM Title	*Subscription Number*	*Collection Kit Number*
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
Transaction Processing and Data Management Redbook Collection	SBOF-7240	SK2T-8038
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
System/390 Redbooks Collection	SBOF-7201	SK2T-2177

How To Get ITSO Redbooks

This section explains how both customers and IBM/Lotus employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL

http://www.redbooks.ibm.com/redbooks.

How IBM and Lotus Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

IBM Employees

- PUBORDER - to order hardcopies in the United States.
- GOPHER link to the Internet - type gopher.wtscpok.itso.ibm.com.
- Tools disks
  - To get LIST3820s of the redbooks listed in the Other ITSO Lotus Related Publications section, type one of the following commands:

    tools sendto ehone4 tools2 redprint get sg24xxxx package

    tools sendto canvm2 tools redprint get sg24xxxx package (Canadian users only)

  Note The ITSO Lotus publications, and this redbook *Guide to Deploying Domino Go Webserver*, are not available in LIST3820 or BookManager format. However, PDF format is available for those books from **http://www.lotus.com/redbook**

  - To get lists of redbooks:

    tools sendto wtscpok tools redbooks get redbooks catalog

    tools sendto usdist mkttools mkttools get itsocat txt

    tools sendto usdist mkttools mkttools get listserv package

  - To register for information on workshops, residencies, and redbooks:

    tools sendto wtscpok tools zdisk get itsoregi 1996

- For a list of product area specialists in the ITSO:

  tools sendto wtscpok tools zdisk get orgcard package

- Redbooks Home Page on the World Wide Web

  http://w3.itso.ibm.com/redbooks/redbooks.html

  Note The ITSO Lotus Related Publications are listed on this site, but not the ITSO Lotus redbooks, which are available in Adobe Acrobat format from **http://www.lotus.com/redbook**

- IBM Direct Publications Catalog on the World Wide Web

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- ITSO4USA category on INEWS
- Online - send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL.

Lotus Employees

- This redbook, and those listed in the ITSO Lotus Publications section of this book, can be ordered from the Internal Orders Database using Lotus part numbers.
- In addition PDF versions of those books can be downloaded from:

  http://www.lotus.com/redbook

- The other ITSO publications listed can be accessed on the Redbooks Home Page on the World Wide Web at:

  http://w3.itso.ibm.com/redbooks/redbooks.html

All Employees

- Internet Listserver

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

IBM Mail	Internet	
In United States	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	bookshop at dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders - send orders to:**

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29554	Calgary, Alberta T2P 3N5	DK-3450 Allerod
Raleigh, NC 27626-0570	Canada	Denmark

- **Fax - send orders to:**

United States (toll free)	1-800-445-9269
Canada (toll free)	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States) or (+1) 415 855 43 29 (Outside USA) — ask for:**
  Index #4421 Abstracts of new redbooks
  Index #4422 IBM redbooks
  Index #4420 Redbooks for last six months

- **Direct Services — send note to softwareshop@vnet.ibm.com**

- **On the World Wide Web**

Redbooks Home Page	`http://www.redbooks.ibm.com/redbooks`
IBM Direct Publications Catalog	`http://www.elink.ibmlink.ibm.com/pbl/pbl`

  The current redbook *Guide to Deploying Domino Go Webserver* is also available in Adobe Acrobat format on the World Wide Web. The URL is `http://www.lotus.com/redbook`.

- **Internet Listserver**
  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service send an e-mail note to annouce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

❑ Please put me on the mailing list for updated versions of the IBM Redbook Catalog.

First name | Last name

Company

Address

City | Postal code | Country

Telephone number | Telefax number | VAT number

❑ Invoice to customer number

❑ Credit card number

Credit card expiration date | Card issued to | Signature

We accept American Express, Diners, Eurocard, MasterCard, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Index

T

U

V

W

X

ITSO Redbook Evaluation

Guide to Deploying Domino Go Webserver SG24-2002-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form at http:www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:

(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes _____ No _____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK)

_____

_____

_____

_____

_____

Printed in the U.S.A.

SG24-2002-00

Part No. 12991

Lotus®

IBM

SG24-2002-00